BATCH FILE TIPS INSIDE ... PAGE 55

DOS RESOURCE GUIDE

The PC Productivity Magazine

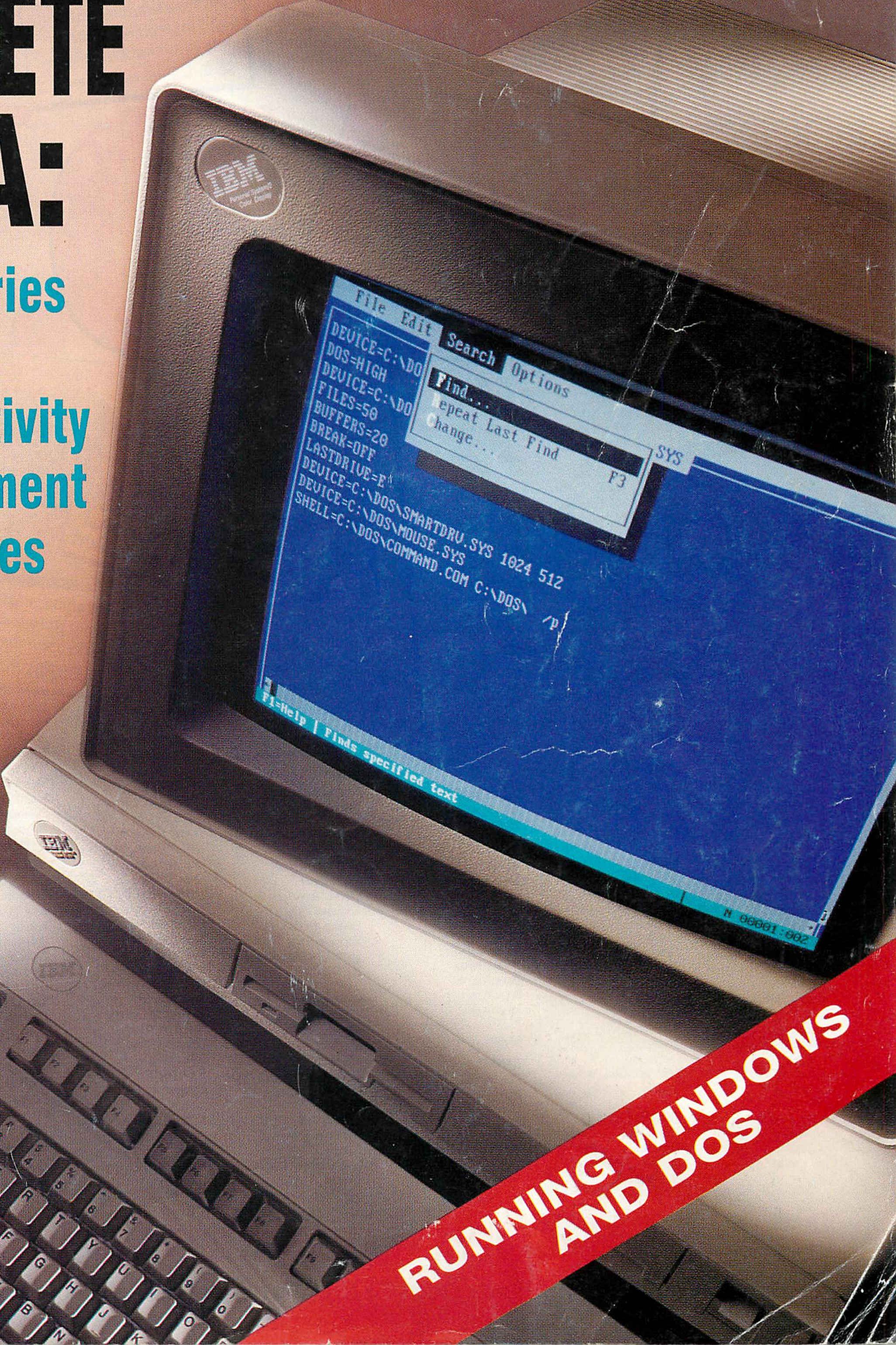
AN IDG COMMUNICATIONS PUBLICATION
NUMBER 4
$4.95
CANADA $5.95
UK £3.00

THE COMPLETE DOS Q&A:

- Files and Directories
- DOS Setup
- Improved Productivity
- Memory Management
- Hardware Upgrades

GET STARTED WITH QBASIC

7 WAYS TO SAFEGUARD YOUR DATA



```
DEVICE=C:\DO
DOS=HIGH
DEVICE=C:\DO
FILES=50
BUFFERS=20
BREAK=OFF
LASTDRIVE=E
DEVICE=C:\DOS\SMARTDRV.SYS 1024 512
DEVICE=C:\DOS\MOUSE.SYS
SHELL=C:\DOS\COMMAND.COM C:\DOS\  /p
```

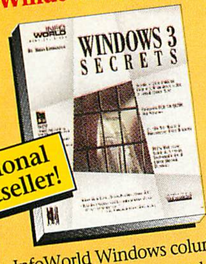
RUNNING WINDOWS AND DOS

AN IDG COMMUNICATIONS PUBLICATION
09
0 74470 78613 9

Look for PC Books *nationwide* at:

B. Dalton
Barnes & Noble
Bookstop/Bookstar
Crown/SuperCrown
Encore
Software Etc.
Waldensoftware
Waldenbooks

Windows 3 Secrets

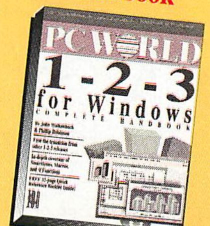


National Bestseller!

InfoWorld Windows columnist reveals undocumented features & workarounds, configuration tips, & 36 top Windows programs.

$39.95 US/$52.95 Canada,
831 Pages, Two 5 1/4" disks

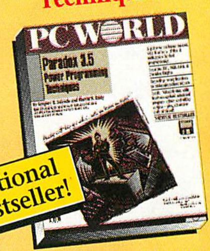
PC World 1-2-3 for Windows Complete Handbook



Definitive reference by two 1-2-3 experts. FREE Quick Reference booklet included.

$29.95 US/$36.95 Canada, 704 Pgs

PC World Paradox Power Programming Techniques



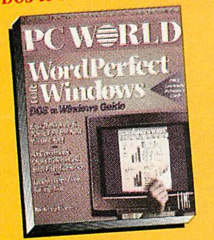
National Bestseller!

Critically acclaimed–your only Paradox bible to PAL & advanced techniques. With disk of programs.

$39.95 US/$52.95 Canada,
784 Pages, 3 1/2" disk

READ.ME

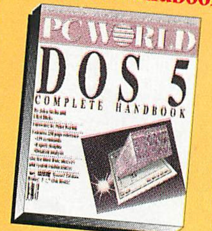
PC World WordPerfect for Windows DOS to Windows Guide



Ideal transition guide for new Windows users. FREE Command Reference Card included.

$19.95 US/$26.95 Canada,
340 Pages

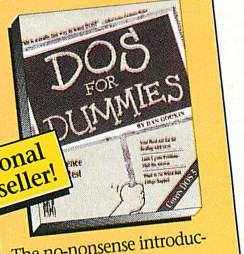
PC World DOS 5 Complete Handbook



Master the new DOS quickly and easily. FREE with purchase: Norton Commander Special Edition disk!

$29.95 US/$39.95 Canada,
696 Pages, 5 1/4" disk

DOS For Dummies



National Bestseller!

The no-nonsense introduction & problem solving guide; covers through DOS 5.

$16.95 US/$21.95 Canada, 224 Pgs

Official XTree MS-DOS & Hard Disk Companion

National Bestseller!

Only official guide to all versions of XTree hard disk software! National bestseller.

$15.95 US/$21.95 Canada, 292 Pgs

ALASKA
Book Cache
All locations,
including Anchorage
Baker and Baker
Booksellers
Fairbanks
Big City Books
Juneau

NO. CALIFORNIA
Capitola Book Cafe
Capitola
A Clean
Well Lighted Place
Cupertino
The Booksmith
San Francisco
BookShop Santa Cruz
Santa Cruz
Booksellers
Campbell
Stacey's Bookstore
Cupertino
Modesto
Palo Alto
San Francisco (all locations)
Tower Books
Chico
Citrus Heights
Concord
Mountain View
Sacramento (all locations)
San Mateo
Whole Earth Access
Concord
San Francisco
San Jose
San Mateo
San Rafael
Cody's Books
Berkeley
Computer Literacy
Cupertino
San Jose
Sunnyvale
Gayle's Books
Davis
UCD Bookstore
Davis
Stanford
Bookstore
Palo Alto
Stanford
Aardvark
Books
San Francisco
Foley's Books
San Francisco
Franciscan
Shops
San Francisco
Copperfield's
Books
Santa Rosa

SO. CALIFORNIA
Huntley
Bookstore
Claremont
UCLA
Bookstore
Los Angeles
USC Trojan Bookstore
Los Angeles
Opamp Technical Books
Los Angeles
Vroman's
Pasadena
San Diego
Technical Books
San Diego
University Bookstore
San Diego
University Bookstore
La Jolla

COLORADO
University Bookstore
Greeley
Tattered Cover
Bookstore
Denver
Colorado Book Store
Boulder

CONNECTICUT
Huntington's Bookstore
Hartford
R.J. Julia Booksellers
Madison
Yale Co-Op
New Haven
Books Unlimited
Rocky Hill
U Conn Co-op
Storrs

DISTRICT OF COLUMBIA
Reiter's
Scientific &
Professional
Books
Reprint
Book Shop
Trouer

FLORIDA
Liberties
Boca Raton
Kingsley's
Sarasota
White's
Orlando

GEORGIA
Oxford Bookstore
Atlanta
Douglasville Books
Douglasville
Micro Center
Marietta

HAWAII
Honolulu Book Shops
Honolulu

ILLINOIS
Pages For All Ages
Champaign
57th Street Books
Chicago
Kroch's &
Brentano's
Chicago
Readmore
Bookworld
Rock Island
Bookworld III
Rockford
Book Center/
Illini Union
Urbana
Lora & Wally Grigg
Woodstock
Books on the Square
Woodstock

IOWA
University of Iowa
Iowa City

INDIANA
Horizon Books
Indianapolis
Book Corner
Bloomington
Book World
Mishawaka
Ball State University
Muncie

KANSAS
Crossroads
Hutchinson
University Book Store
Manhattan

MAINE
Harbour Books
Yarmouth

MARYLAND
Sidney Kramer Books
Rockville
Maryland Book Exchange
College Park

MASSACHUSETTS
Words Worth
Cambridge
Charlesbank at BU
Boston
Brookline Booksmith
Brookline
Barillari Books
Cambridge
Quantum Books
Cambridge
Baker Books
New Bedford
The Open Book
Westboro
Waterstone's
Boston
Softpro
Burlington

MICHIGAN
Metro News Center
Birmingham
Book World
Grand Rapids
Little
Professor Book Center
Dearborn
Young & Welshans
Flint
Bookworm Eastbrook Mall
Grand Rapids
MacGregor News
Mt. Pleasant
O'Neill & Austin
Rochester Hills
Fireside Bookshop
Rochester Hills
James Lundberg Bookstore
Big Rapids

MINNESOTA
Baxter Books
Minneapolis
Maverick Bookstore
Mankuto
Odegard Books Saint Paul
St. Paul

NEBRASKA
Nebraska Bookstore
Lincoln
University Bookstore
Lincoln

NEW HAMPSHIRE
Dartmouth Bookstore
Hanover

NEW JERSEY
McGraw Hill
Hightstown
Village Comp. & Software
Cedar Knolls
Simplex Comuter. Center
Kenilworth

NEW YORK
Barnard Book Forum
New York City
Classic Bookshop
NYC (all locations)
Coliseum Books
New York City
McGraw-Hill Bookstore
New York City
Papyrus Booksellers
New York City
Tower Books
NYC (all locations)
J & R Computer World
New York City
47th Street Computer
New York City
Computer Store
Forest Hills
Book Revue
Huntington

NORTH CAROLINA
Atticus Books
Greensboro
News 'n Novels
Greensboro

OHIO
Wilkies University Shoppes
Fairborn
Wit & Wisdom
Beachwood
Rocky River
Ohio State University
Columbus
Books & Co.
Dayton
Wilkie's South
Dayton
Micro Center
Columbus
Mayfield Heights
Westerville

OREGON
Oregon State Univ.
Corvallis
Book Mark
Eugene
Powell's Bookstore
Beaverton
Powell's Technical Books
Portland
Reed College
Portland

PENNSLYVANIA
The Book Store
Philadelphia
The Book Center
Pittsburgh
Business & Computers
Willow Grove

TENNESSEE
Zimmerman's Books
Johnson City
Bookworld
Nashville
Campus Bookstore
Knoxville

TEXAS
PRNT Bookstore/
InfoMart
Dallas
Sam Houston
Book Shop
Houston
University
Co-Op
Austin
Bookstop
All locations
Taylors
Tech. Books
Dallas
(all locations)
Ft. Worth
Plano
J.A. Majors
Tech. Books
Houston

UTAH
BYU Bookstore
Provo

WASHINGTON
Tower Books
Bellevue, Seattle
University Bookstore
Bellevue
Elliott Bay Book
Seattle
University Book Store
Seattle

WISCONSIN
Conkey's Book Store
Appleton
Harry W. Schwartz
Milwaukee

CANADA

ALBERTA
The Book Company
Calgary
Calgary Computer Books
Calgary
Audrey's Books
Edmonton
Greenwood's Bookshoppe
Edmonton
Software House (West)
Edmonton

BRITISH COLUMBIA
Unihall Computer Books
Vancouver
Granville Book Co.
Vancouver
Siliconnections Books
Vancouver
Bolen Books
Victoria

NOVA SCOTIA
The Book Co. Entitlement
Halifax

ONTARIO
Coles Book Stores
(all locations)
Lichtman's Stores
Toronto (5 locations)
Thornhill
The Book Company
Ottawa
Odyssey Comp. & Business
Ottawa
Albert Britnell
Toronto
Books For Business
Toronto
World's Biggest
Bookstore
Toronto
University of Toronto
Toronto
University of Waterloo
Waterloo

QUEBEC
Camelot-Info Computer
Montreal
Roswell Computer Books
Montreal

SASKATCHEWAN
Book & Brier Patch
Regina

IDG
BOOKS

IDG Books Worldwide
155 Bovet Road, Suite 610
San Mateo, CA 94402
(415) 312-0650

Buy Me At These Locations Or Call 1-800-762-2974

Ask for IDG Books, BK=PCT3. Phone orders 7-5 EST. You can also call IDG Books at (415) 312-0650 (9-5 PST) to order direct.

Jonathan Young

DOS RESOURCE GUIDE
The PC Productivity Magazine

editor
Michael Comendul

managing editor
Marilyn McMaster

copy editor
Beth Jala

contributing editor
John Wolfskill

contributing writers
Harry Bee, Hardin Brothers, Brett Glass, James Langdell, Brian Livingston, Jack Nimersheim, Max Ribble, Larry Sessions

manufacture and electronic publishing
Lou Ann Morin
Lindy Palmisano

Back issues of *DOS Resource Guide* are available for $5.50, postage paid. Call: 800-343-0278 (9 a.m. to 5 p.m. EST).

How to contact us: *DOS Resource Guide* welcomes your comments, complaints, DOS tips, and article submissions. Submit them in care of our offices by mail, through CompuServe (75300,2361), or through MCI Mail (469-7260).

DOS Resource Guide (ISSN–1506–7364) is published quarterly by Business Computer Publishing Inc. Business offices at 80 Elm St., Peterborough, NH 03458. Editorial and circulation offices: 603-924-0100. Entire contents copyright 1992 by Business Computer Publishing Inc. No part of this publication may be printed or otherwise reproduced without written permission from the publisher. *DOS Resource Guide* assumes no responsibility for damages due to errors or omissions.

BUSINESS COMPUTER PUBLISHING INC.

president/publisher
Jeff DeTray

editorial director
Michael Comendul

IDG COMMUNICATIONS *DOS Resource Guide* is a publication of International Data Group, the world's leader in information services on information technology.

We're Listening

We've heard from dozens of readers since starting the *DOS Resource Guide*, many with praise, some with criticism, but almost all with questions about this cryptic operating system. We appreciate all of these comments.

Your letters, especially your DOS experiences are important to the editors as we try to shape future issues of *DOS Resource Guide*. For example, in this issue we've added more material about DOS's inner workings, and we've tried to divide the magazine into sections. Each section of the magazine starts with a fundamental article followed by a Q&A session. All of our authors are available (by mail) to answer your questions. Take advantage of their experience.

We've also added a QBasic programming section, the language that comes with MS-DOS 5.0, and a Windows Q&A section.

At every turn, *DOS Resource Guide* tries to simplify the language of computing, but, unavoidably, jargon creeps into our text. For example, in this issue a number of writers mention XT or an AT computers. These are references to classes of IBM-compatible computers, each powered by a different chip, or CPU (central processing unit). The original IBM PCs and XTs were powered by 8088 or 8086 chips designed by Intel Corp. In addition, the XT (distinct from a PC) introduced a hard drive as standard equipment. The AT was driven by Intel's more powerful 80286 chip and defined what has become known as the AT-bus standard, a common design for the expansion slots in a computer. These slots are used to expand your computer by adding internal hardware—a video board, for example. But these computers are dated. They are giving way to 386SX computers, and now, 486 computers—computers driven by 80386SX chips or 80486 chips, respectively.

This is but one example of unavoidable jargon. If the language in *DOS Resource Guide* gets too technical, let us know. And, when you call or write our office with a problem, please be patient. We'll try our best to answer your questions, and if we can't help, we'll try to connect you with someone who can.

Just as important as your criticism and questions are the DOS tips, shortcuts, and batch files that you may be using every day. In future issues, we will print as many of our readers' tips as space allows, and for the best of these reader-submitted tips, *DOS Resource Guide* will pay from $25 to $50. Just one of the small rewards of DOS computing.

Let us know what you think of the changes.

Michael J. Comendul

DOS RESOURCE GUIDE
The PC Productivity Magazine

NUMBER 4

DOS FUNDAMENTALS

Of Files and Directories *by James Langdell* 9
All you need to know to create, store, recall, and transform files.

Q&A: Files and Directories 13

DOS Goes Graphical *by Jack Nimersheim* 15
Version 5 .0's DOSSHELL offers something for novices and experienced users alike.

Managing DOS Setup Files *by Hardin Brothers* 19
Someday, bootup problems are going to come. Be prepared!

Q&A: Setting Up DOS 26

IMPROVING PRODUCTIVITY

A Six-Pack of DOS Treasures *by John Wolfskill* 29
A half-dozen gems that can make your computing life easier.

Q&A: Improving Productivity 35

MEMORY MANAGEMENT

Sorting Out Memory *by Brett Glass* 38
What every computer user needs to understand about expanded and extended memory.

Q&A: Memory Management 42

HARDWARE

On the Upgrade Path *by John Wolfskill* 45
A change of hardware components can give an aging computer a new lease on life.

Q&A: Hardware 48

Boosting Your PC's Memory *by David S. Veale* 51
How to select and install additional system memory.

BATCH FILE MAGIC

Take Control with Batch Files *by Harry Bee* 55
Writing batch files is easy, but challenging enough to be fun. These four time-saving programs offer you a perfect opportunity to take up the challenge.

PROGRAMMING

Elementary Steps to Programming *by Hardin Brothers* 63
Create a handy phone directory while educating yourself about the QBasic programming language bundled with DOS 5.0.

Seven Programs to Safeguard Your Files *by Larry Sessions* 69
Protect valuable data against accidental loss—and prying eyes.

WINDOWS

Q&A: Windows 3.0 *by Brian Livingston* 75

Tips:

#1 Using CONFIG.SYS to increase the size of your environment space 14
#2 Floppy disk labels that stick 18
#3 Ask a DOS 5.0 question, get a quick answer 18
#4 DIRCMD can do good things for directory listings 34
#5 The mysteries of NUL revealed 44
#6 Team up DIR with DEL to safely erase files 62
#7 Removing extraneous files from your root directory 62
#8 Make DOS 5.0's shell into a true graphics display 68
#9 Putting the ? wildcard to good use 80

Letters to the Editors

DOS Gold

I would like to thank you ever so much for bringing out a magazine that I, as a beginner in the IBM-compatible world, can understand. I've purchased many magazines over the years, and *DOS Resource Guide* is like a bible to me. I read it from cover to cover and experiment with it. You have a good thing going, and I would like to see many more issues come out in the future.

Keep up the excellent magazine; it is worth its weight in gold to me.

Sharon Reddrop
Kingston, Ontario

I recently purchased your new magazine, *DOS Resource Guide*, and enjoyed it very much. I have tons of books on DOS, but I could never grasp the jargon. I was in the process of buying another book when I decided to browse through the magazine section.

I came across your magazine and started to read it. It's embarrassing to say, but I was so engrossed that my husband had to buy the magazine to get me out of the store.

I went home and tried almost every example in the magazine. My favorite is the batch file CURRENT.BAT. I'm always on the computer with my word processing program (my newest software package) to learn as much as I can. I use CURRENT.BAT, and it's great!

Thank you for creating this magazine just for me.

Diane Applegate
Madison, N.J.

Your magazine needs to be the replacement for the *Microsoft MS-DOS User's Guide and Reference Manual.* I picked up the third issue of *DOS Resource Guide*, and I haven't put it down yet. I have learned more about DOS through this magazine than I have learned in two manuals, three different computer magazines, and the people I work with.

I bought my first PC in February 1991. I've edited my AUTOEXEC.BAT and CONFIG.SYS files so much that I'm beginning to rub a hole in my hard drive. I've had your magazine for a week now, and I have finally optimized my system.

Mark H. Saunders
Simpsonville, S.C.

Thank you for your recent issue of *DOS Resource Guide*, "The Essential Guide to DOS." It is the most valuable 80 or so pages of DOS help I have found to date. The article called "Memory Management Made Easier" was particularly helpful. You cleared up in a few pages several questions that still remained after I read numerous other articles.

Charlie Clark
Leona Valley, Calif.

Saw a copy of *DOS Resource Guide* Number 3 in the supermarket magazine rack, perused it, bought it, and was immensely impressed. It obviously is prepared and edited for people like me: a PC advocate who is not a "buff" and does not have an advanced degree in computer engineering.

Howard M. Lane
Melbourne, Fla.

I recently bought the premiere issue of *DOS Resource Guide*. As I read through it, I was impressed by the quality of the articles and the useful tips the magazine contained. My only disappointment was that when I began to hunt for the subscription card, I couldn't find one. The copy I bought at the bookstore was the last one on the shelf, so I wasn't able to look through another copy to find one.

Needless to say, I am very interested in obtaining a subscription to this wonderful magazine. Please rush me whatever I need to fill out to become a subscriber.

Michael W. Beard
Long Beach, Calif.

Sorry, Mr. Beard, DOS Resource Guide, *a quarterly, is not yet offered for subscription. Keep an eye on that newsstand.—eds.*

As a relatively new PC user, I'd like to let you know how happy I was to come across your *DOS Resource Guide* recently. The information and the way in which it is presented has really unlocked the mysteries of MS-DOS for me.

I'm currently studying up on DR DOS 6.0 in preparation for installation. From what I've read and understand so far, I should be able to apply your ideas to either brand of PC operating system. From reading the top 10 software sales charts, I assume that a lot of other people are using DR DOS also. I hope you won't be like other major publications and ignore this product. I noticed that in an article on memory managers in the February 11, 1992 issue of *PC Magazine* DR DOS 6.0 gets better overall ratings than MS-DOS 5.0 and even gets a higher score for Windows support than does MS-DOS 5.0!

Thanks for being there with your magazine. This is one novice who is getting more skilled every day, thanks to you.

Ivan Sinkler
Alameda, Calif.

Food for Thought

By a stroke of luck, the only copy of *DOS Resource Guide* to reach the shelves of our local (and small) store was gleaned by me, a solitary PC consultant of six years.

While I can (inwardly) boast that I know DOS very well, I never seem to find the time to write DOS code for myself, and your very excellent magazine has stirred many fresh ideas. Virtually all of the guide's articles are of interest to me and are written at a level that I can easily and quickly comprehend.

No doubt there is no shortage of material to discuss, but a few areas still mystify me. Perhaps you would address some of the following in future issues:

Keyboards. They still represent black magic to me! I don't understand how or why the Num Lock and Scroll Lock lights come on at bootup, or more importantly, how to turn them off at bootup. On some keyboards, the left Ctrl key turns on Caps Lock. How can I control these problems?

Environment. While I generally understand this, many people don't. Many of my clients have earlier 286 machines that don't conform to the EMS 3.2 or LIM EMS 4.0 specs, and with all the hype about Windows, I have to explain that not everyone can easily extract the full power of Windows. In fact, I have yet to see a concise explanation of normal, real, and protected modes; extended and expanded memory; EMM386 and XMA2EMS; and installing Windows 286 or Windows 3.0 in a non-LIM EMS environment. I would like to see an article on these topics that explains how conventional memory is affected. With all the hype over graphical interfaces, this kind of article is long overdue.

DOS error messages. Despite making a valiant effort, I have never been able to obtain a *complete list* of DOS error messages. I do have one that must be five years old, but, clearly, it must no longer be valid, except for the most basic of calls.

Escape codes. I would like to see something on them, their applications, and how they're generated, including the actual keystrokes.

Also, I have never seen all (27?) internal DOS commands listed at once. Perhaps you could list these by version.

Perhaps a good idea for a monthly column might be to expand on a single DOS command or feature in batch file programming—functions such as piping (|), redirection (< and >), and the use of percent signs (%). These often are misunderstood, at least by me.

Hiding a file is no trick, but how can you hide a whole directory?

Another poorly documented area involves the switches used in conjunction with many DOS calls. Some are common. Some are unique. Seeing them all together in one place might put them in perspective. As you have pointed out, some switches—such as the ones for CHKDSK—are not well documented.

I could go on indefinitely, but I have work to do.

Gordon A. Clark,
President
BRANTA Systems
Montreal, Quebec

Good suggestions all. You'll find our first try at resolving the Num Lock problem re-explained (without the original error) elsewhere in this section.—eds.

Omissions and Corrections

Great idea for a mag! At long last, someone is recognizing that I don't get my kicks knowing how all the little green men in my 286 do their jobs. I just want to be able to tell them what to do and then be certain they'll do it.

So, when I was browsing at the local magazine rack, I got excited over your tip called "Nixing Num Lock" (Number 1, page 13). I even bought the mag.

I *carefully* replicated the script provided by you. Then, I typed:

```
TYPE RESETNL.SCR
```

pressed Enter, and the file was displayed correctly. I typed:

```
DEBUG<RESETNL.SCR
```

pressed Enter, and got an error message.

I think sticking it to the dreaded Num Lock is such a great idea that a lot of readers who might have had their assembly of the program nixed would like a second look at this one.

What can you do for us?

S.M. Steenhaus
Mission, British Columbia

I purchased a copy of your premiere issue, and the first thing that I tried was "Nixing Num Lock." I tried it using my word processor (Word 5.0) and saving it in ASCII format. When I attempted to convert RESETNL-

.SCR to RESETNL.COM using Debug, I got an error message.

Where is the goof?

Bernard W. Weiss
Cranbury, N.J.

Somewhere in the editing and production process, a colon was omitted from the sixth line of the listing. Without the comments—which should not be typed when you enter the listing into your text editor—RESETNL.SCR should read as follows:

```
n resetnl.com
a           100
xor         ax,ax
mov         es,ax
mov         al,20
es:
xor         [417],al
int         20

rcx
0d
w
q
```

Be sure to differentiate between the letter o and the number zero. For instance, the line following RCX is 0d, not od.—eds.

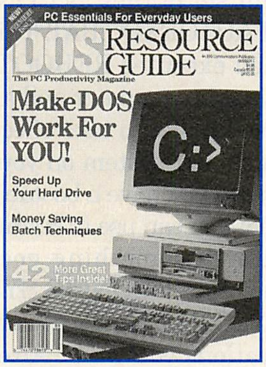
DOS RESOURCE GUIDE welcomes feedback from its readers. Address your DOS questions or comments to:

DOS Resource Guide
80 Elm St.
Peterborough, NH 03458
603-924-0100
603-924-9384 (fax)

You also may contact the editors electronically on MCI (469-7260) or CompuServe (75300-2361).

DID YOU EVER NOTICE THAT C:> DOESN'T REALLY POINT ANYWHERE?

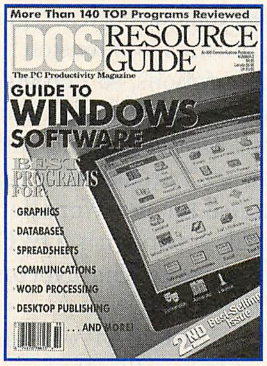
If you need DOS direction, get into the only quarterly devoted to its inner workings: DOS RESOURCE GUIDE. Each issue is packed with short-cuts, how-tos, and solid explanations of DOS's most intimate workings.


PC Essentials For Everyday Users
DOS RESOURCE GUIDE
Make DOS Work For YOU!
Speed Up Your Hard Drive
Money Saving Batch Techniques
42 More Great Tips Inside!

DOS RESOURCE GUIDE #1

Make DOS Work for You

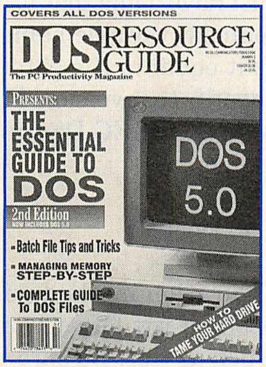
Covers six areas of computing that are vital to DOS users: start-up files (AUTOEXEC.BAT and CONFIG.SYS), batch file creation, hard drives, safe backups, organizing file management, and hardware maintenance. Includes: List'em, a batch file that helps you keep appointments; 10 tips to keep your hard drive healthy; speed-up tips for a dogged hard drive; a batch file that automates backups; DOS shortcuts, hardware dos and don'ts. Plus 42 provocative pointers in DOS.



DOS RESOURCE GUIDE #2

Guide to Windows Software

A compendium of software briefs that tells you what's up in the world of Windows. (Even a DOS user has to keep up with the competition.) Guide #2 briefly reviews over 140 programs for Microsoft Windows including word processors, communications programs, financial and mathematical programs, management programs, developer's tools, and graphics software. Complete list of manufacturers.


COVERS ALL DOS VERSIONS
DOS RESOURCE GUIDE
PRESENTS: THE ESSENTIAL GUIDE TO DOS
2nd Edition
DOS 5.0
Batch File Tips and Tricks
MANAGING MEMORY STEP-BY-STEP
COMPLETE GUIDE To DOS Files
HOW TO TAME YOUR HARD DRIVE

DOS RESOURCE GUIDE #3

The Essential Guide to DOS

Who are we to argue with success! Our second edition of this classic is updated to include information on DOS 5. Guide #3 includes an annotated list of DOS files and when to use them; articles that will change your view of CHKDSK, COPY, and XCOPY; a cluster of batch file assists; and some clear-headed advice on organizing files on your hard drive. Plus 14 titillating tips in DOS.

ORDER YOUR BACK ISSUES TODAY!

Each issue of DOS RESOURCE GUIDE is **$5.50** postage paid

Call 1-800-343-0728 (9AM to 5PM EST) In New Hampshire dial 924-0100

OR WRITE: DOS RESOURCE GUIDE
Back Issues
80 Elm St.
Peterborough, NH 03458

DOS Tips From Readers

After creating my own versions of DOS 5.0 batch files to change my system's configuration settings automatically, I noticed the article "Boot Disk for Every Occasion" (Number 1, page 4). It advises that you keep different sets of CONFIG.SYS and AUTOEXEC.BAT files for handling different situations. My approach makes it easier to change settings .

Without pre-established sets of configuration files, you might have to switch to the "vanilla" setup on your bootable floppy disk. On top of that, you'll have to locate the floppy, insert it in the drive, and type Ctrl-Alt-Del. Once the reboot brings up a prompt, you then must consider other details, such as whether your PATH statement is adequate and whether you need ANSI.SYS (if it isn't in the configuration file on your floppy disk).

After you finish the work that needs to be done under the new configuration, you must remove your floppy disk and reboot by pressing Ctrl-Alt-Del again to return to the previous configuration.

My system simplifies all this, but to follow my suggestions, you need REBOOT.COM, a Debug script that lets you reboot your computer without touching a key or computer switch (see "Creating REBOOT.COM").

Instant Configuration Files

Once you have REBOOT.COM in hand, you can begin to create a simple menuing system to automate the selection of different setup configurations. After preparing and testing your configuration files, you store them in your root directory as individually named file sets. Changing from one configuration to another involves invoking a batch file to rename the active CONFIG.SYS and AUTOEXEC.BAT files and renaming a different set of files as CONFIG.SYS and AUTOEXEC.BAT. This batch file then executes REBOOT.COM to restart the computer and allow the new setup files to take effect.

To create a similar system on your machine, be sure you have a bootable floppy disk that you can use to restart your computer if something goes wrong. Then create a CONFIG.SYS and AUTOEXEC.BAT file for each of the configurations you want. After testing the files, rename each set, giving each set a mnemonic extension. I need four configurations, so I created four sets of configuration files. Those for my regular DOS session end with .DS1, those for my "bare bones" setup with .BB1, those for my WordPerfect/Grammatik setup with .DWG, and those for running The Norton Utilities' NDOS with .NU1.

Each AUTOEXEC.BAT version you create must have a line that uses a DOS environment variable named FIG and sets it equal to the extension used to identify that version. For example, my AUTOEXEC.DS1 file contains this line:

```
SET FIG=DS1
```

My reconfiguration method will not work properly if you forget to include a line such as this. Be sure to place a SET command at the end of each AUTOEXEC.BAT version you create. This command will display important information to you just before displaying the DOS prompt. Among other things, the SET command displays the three-letter identifier stored in FIG by the active version of AUTOEXEC.BAT.

The exact content of your setup files depends on your needs and preferences. You can have as many sets of configuration files as you need. Just be sure to give each set a unique extension and to store all the sets in the root directory of your hard drive.

(Note that my file sets include a program called NSTART.BAT because one of my configurations activates Norton's NDOS.COM. When I use this setup, Norton executes CONFIG.SYS and looks for NSTART.BAT, executing it before executing AUTOEXEC.BAT.)

The Batch Files

Next, you must create short batch files similar to my batch files D.BAT, B.BAT, G.BAT, and N.BAT (see Program Listings 1, 2, 3, and 4). These set the DOS environment variable FYG equal to the extension you use on the corresponding configuration files. Then, FYG is compared with FIG to see if the requested setup is in effect.

You'll need a short batch file for each set of configuration files you create. Save them under single-letter filenames with .BAT extensions. When you create MENU.TXT, the menu display for my configuration system (see Program Listing 5), you can use the individual letters as menu choices. Don't use *C* because I use C.BAT to start the entire reconfiguration system.

Now, make a \CONFIG directory on your hard drive. In it, store the short batch files you've just created, along with MENU.TXT and DOSWITCH.BAT, NUSWITCH.BAT, NOCHANGE.BAT, MENULOOP.BAT, MISSA.BAT, MISSC.BAT, MISSN.BAT, and FIGMENU.BAT (see Program Listings 6 through 13).

Program Listing 1. D.BAT.

```
@ECHO OFF
SET FYG=DS1
IF “%FIG%″==″%FYG%″ NO CHANGE
DOSWITCH
```

End

Program Listing 2 B.BAT.

```
@ECHO OFF
SET FYG=BB1
IF “%FIG%″==″%FYG%″ NO CHANGE
DOSWITCH
```

End

Program Listing 3. G.BAT

```
@ECHO OFF
SET FYG=DWG
IF “%FIG%″==″%FYG%″ NO CHANGE
DOSWITCH
```

End

Program Listing 4. N.BAT.

```
@ECHO OFF
SET FYG=NU1
IF “%FIG%″==″%FIG%″ NO CHANGE
NUSWITCH
```

End

Program Listing 6. DOSWITCH.BAT.

```
@ECHO OFF
CD\
IF NOT EXIST CONFIG.%FYG% MISSC
IF NOT EXIST AUTOEXEC.%FYG% MISSA
IF EXIST CONFIG.SYS REN CONFIG.SYS CONFIG.%FIG%
IF EXIST AUTOEXEC.BAT REN AUTOEXEC.BAT AUTOEXEC.%FIG%
IF EXIST NSTART.BAT REN NSTART.BAT NSTART.%FIG%
REN CONFIG.%FYG% CONFIG.SYS
REN AUTOEXEC.%FYG% AUTOEXEC.BAT
REBOOT
```

End

Program Listing 7. NUSWITCH.BAT.

```
@ECHO OFF
CD\
IF NOT EXIST AUTOEXEC.%FYG% MISSA
IF NOT EXIST CONFIG.%FYG% MISSC
IF NOT EXIST NSTART.%FYG% MISSN
IF EXIST CONFIG.SYS REN CONFIG.SYS CONFIG.%FIG%
IF EXIST AUTOEXEC.BAT REN AUTOEXEC.BAT AUTOEXEC.%FIG%
IF EXIST NSTART.BAT REN NSTART.BAT NSTART.%FIG%
REN CONFIG.%FYG% CONFIG.SYS
REN AUTOEXEC.%FYG% AUTOEXEC.BAT
REN NSTART.%FYG% NSTART.BAT
REBOOT
```

End

Program Listing 8. NOCHANGE.BAT.

```
@ECHO OFF
@ECHO You are already in the %FYG% configuration.
MENULOOP
```

End

Program Listing 9. MENULOOP.BAT.

```
@ECHO OFF
@ECHO To return to the configuration menu...
PAUSE
CD\CONFIG
FIGMENU
```

End

Program Listing 10. MISSA.BAT.

```
@ECHO OFF
@ECHO There is no AUTOEXEC.%FYG% in the ROOT directory
@ECHO for the requested configuration. Create it to
@ECHO use the requested configuration. Also prepare
@ECHO the CONFIG.%FYG% and NSTART.%FYG% if missing for
@ECHO the requested configuration.
MENULOOP
```

End

Program Listing 11. MISSC.BAT.

```
@ECHO OFF
@ECHO There is no CONFIG.%FYG% in the ROOT directory
@ECHO for the requested configuration. Create it to
@ECHO use the requested configuration. Also prepare
@ECHO the AUTOEXEC.%FYG% and NSTART.%FYG% if missing for
@ECHO the requested configuration.
MENULOOP
```

End

Program Listing 12. MISSN.BAT.

```
@ECHO OFF
@ECHO There is no NSTART.%FYG% in the ROOT directory.
@ECHO for the requested configuration. Create it to
@ECHO use the requested configuration. Also prepare
@ECHO the AUTOEXEC.%FYG% and CONFIG.%FYG% if missing for
@ECHO the requested configuration.
```

End

Program Listing 13 . FIGMENU.BAT.

```
@ECHO OFF
CLS
VER
CD\CONFIG
TYPEMENU.TXT
```

End

Program Listing 14. C.BAT.

```
@ECHO OFF
CD\CONFIG
FIGMENU
```

End

CREATING REBOOT.COM

REBOOT.COM is reprinted with permission from MS-DOS Batch Files *by Kris Jamsa (Microsoft Press, Second Edition, ISBN 1-55615-338-4).*

Simpson's batch files use a small program called REBOOT.COM to restart your computer just as if you had pressed Ctl-Alt-Del. Using a text editor and DOS's Debug, you can create your own copy of REBOOT.COM by carefully performing the following steps:

- Using Edlin or Edit (Edlin comes with all versions of DOS; Edit comes with DOS 5.0) or a word processor that can save files in text or ASCII format, type in the following program listing. This creates and saves the program as REBOOT.SCR.

```
n reboot.com
a 100
mov  ax,40
mov  ds,ax
mov  ax,1234
mov  [72],ax
jmp  ffff:0

rcx
10
w
q
```

The file you created, REBOOT.SCR, is an example of a Debug script. In this case, the script is a set of assembly language statements that can be "assembled" by Debug into an executable machine language program you can run by typing its name at the DOS prompt.

As you type in the listing, please note the difference between the letter *o* and the number zero. Also, be sure to enter a blank line as the eighth line and to press Enter at the end of each line, including the last one. Save the listing as a text or ASCII file. Edlin and Edit always save files in the correct format, but your word processor may require you to select a text or ASCII option.

- Copy REBOOT.SCR into the directory that contains Debug and change to that directory.
- At the DOS prompt, type :

```
debug < reboot.scr
```

When you press Enter, Debug should produce a display that looks something like this:

```
-n reboot.com
-A 100
175C:0100 mov  ax,40
175C:0103 mov  ds,ax
175C:0105 mov  ax,1234
175C:0108 mov  [72],ax
175C:010B jmp ffff:0
 175C:0110
-rcx
CX 0000
:10
-w
Writing 00010 bytes
-q
```

When you type DIR to see a directory listing, you should find that the program REBOOT.COM is there.

- Test the program you have just created by typing REBOOT at the DOS prompt. Your computer should restart just as if you had pressed Ctl-Alt-Del. You can keep REBOOT.COM in your DOS directory or in a directory listed your AUTOEXEC.BAT's PATH statement.

—Jeff DeTray

NOCHANGE.BAT is executed if you try to change to a configuration that is active. NOCHANGE.BAT gives you an error message and executes MENULOOP.BAT to bring up the configuration menu again. MISSA.BAT, MISSC.BAT, and MISSN.BAT are used to display an error message if the proper CONFIG.SYS, AUTOEXEC.BAT, or NSTART.BAT file is missing. Then MENULOOP.BAT brings up the setup menu.

Program Listing 5. MENU.TXT is a text file that FIGMENU.BAT displays on screen to let you know what your configurations are.

```
Select a Configuration

Available Configurations:

(D)  DOS 5.0 Only
(N)  NDOS and DOS 5.0 (NDOS.COM Loaded)
(B)  "bare bones" in DOS 5.0
(G)  DOS - WordPerfect - Grammatik IV

Type one (letter) and press ENTER:
```

Whenever MENULOOP.BAT is executed, it returns you to the \CONFIG directory and executes FIGMENU.BAT to display the menu.

MENU.TXT is a simple text file that FIGMENU.BAT uses to display the menu on screen. As I mentioned above, the menu choices correspond to the names of the short batch files with the single-letter names created earlier. You can create MENU.TXT with the same text editor you use to write the batch files.

The last step is to create a short batch file called C.BAT (*C* stands for *configure*) and put it in a directory listed in the PATH statement of each of your AUTOEXEC.BAT files (see Program Listing 14). If you are in doubt about where to put C.BAT, place it in your root directory.

To switch configurations, just press *C* at the DOS prompt and press Enter. The configuration menu (MENU.TXT) will be displayed. Select one of the single letter choices and press Enter. In a few moments, the configuration files will be renamed and REBOOT.COM will restart your computer, activating the configuration you have selected.

Jerry D. Simpson
Madison, Ala.

Of Files And Directories

This small collection of DOS commands provides all the magic you need to create, store, recall, and transform files.

by James Langdell

Working with files stored on disk is a bit like sorcery. Just as a wizard who learns the name of a mystical being can summon it, transform it, and dismiss it, DOS users who learn the name of a file and the place it lives—a directory—have the power to copy it, rename it, move it, or delete it.

Fortunately, mastering the mysteries of files and directories in MS-DOS requires neither years of study nor arduous tests of skill. It's a simple matter of learning how to discover the names of files, provide effective names for the files and directories you create, and manipulate files once you name them.

What's in a Filename?

Files can hold information of various types. A file might, for instance, have the ready-to-run code of an application program, such as a word processor or a game. It might have the text of a letter you wrote with a word processing program, a drawing you sketched with graphics software, or a digital recording of your voice singing a song. Or it might contain instructions for a program, such as a list of definitions for function keys you choose to use when a particular program is running.

Filenames may contain a maximum of 11 characters. The first part of the filename may have as many as eight characters; the optional extension to the filename may include up to three additional characters. When you use an extension in a filename, DOS expects you to separate the two different parts of the filename with a period.

Usually, the first part of a filename identifies the purpose of the file, and the extension tells what type of file it is. Certain extensions instruct DOS to treat the files with those extensions in special ways. For example, an .EXE, .COM, or .BAT extension indicates that a file contains a program that DOS can execute. If a file doesn't contain one of these extensions, DOS won't be able to run the program.

Some applications have built-in rules governing the assignment of filename extensions. A word processor, for example, might require that all of its document files be given a .DOC extension and all backup copies of those same documents be identified by a .BAK extension. In other cases, you may freely create extensions that help you identify and organize your ever-growing collection of valuable files.

You may assign filenames that include letters, numerals, and most symbols. When it comes to letters, DOS doesn't distinguish between upper- and lowercase letters: Whether you type a filename as HOMEWORK.DAT, homework.dat, or HomeWork.DAT, it's the same file as far as DOS is concerned. Most of the time, DOS displays filenames in uppercase letters.

Characters that have special meaning

for DOS can't be used in filenames, however. For example, your filenames may not include asterisks, colons, slashes (neither / nor \), or question marks, and you can't use periods, except for the one that separates an extension from the rest of the filename.

Nor can you include spaces in filenames. To get around this limitation, you can substitute an underscore character, as in BIG_BIRD.PIC.

Discovering a Filename

When you're looking for the names of files you have available, the handiest tool is DOS's DIR command. When you enter this command by itself, you receive a list of the files in the current directory of the current drive. The first parts of the filenames are in the far left column on the screen; the next column is for file extensions. (The entry for the second column is blank if the file wasn't given an extension when it was created.) DOS displays spaces between the columns; however, when you type a two-part filename at the DOS prompt, you must join the two parts with a period.

Wildcards let you deal with files whose names you don't yet know or manipulate groups of files with a single command.

Other columns in the DIR command's report show the number of bytes in the file and the date and time the file was created (or the last time its contents were modified). If your directory contains so many files that all of them won't fit on the screen at once, they will stream off the top of the screen before you have a chance to read their names. You can make long directory listings such as these more manageable in two ways. If you want to see a full report for each file, one screen at a time, simply type in the following:

```
DIR /P
```

(Think of the /P parameter as shorthand for *page by page.*) Or, if you're interested only in looking through a list of the filenames, type:

```
DIR /W
```

(Think of /W as shorthand for *wide.*) DOS will display the filenames in a compressed format that lists several names on a line.

If you were disappointed earlier when I mentioned that DOS doesn't let you create filenames that included such amusing characters as the question mark and asterisk, you will be more than compensated by the power these two symbols give you over files. They let you deal with files whose names you don't yet know or manipulate a group of related files within a directory by typing a single command.

The question mark and asterisk are known as wildcard characters. When you type a command that includes them, these characters act as placeholders for characters in a filename. You can substitute a question mark for any individual character in a filename. If a directory includes several files with names such as CHAP02, for example, you can produce a list of only these files by entering this command:

```
DIR CHAP??
```

This command will list the files you requested and ignore those with similar names that contain more than six characters, such as CHAPTER7 or CHAP12.BAK.

The asterisk takes the place of a group of characters in a filename or extension. For example, you can list all the files in a directory with names beginning with the letters *CH* by entering the following:

```
DIR CH*
```

Or, you could list all the backup files that have a .BAK extension in their filenames by entering this command instead:

```
DIR *.BAK
```

When you invoke the DIR command, you might notice something puzzling. Some filenames are followed by the expression *<DIR>* instead of by an indication of file size. Or, if you enter the DIR /W command with DOS 5.0, some names are surrounded by brackets. These distinguishing marks identify a directory, which sometimes is home to an additional group of files.

You will explore more about files and their names later in this article, but the key to mastering files is knowing how to track them down in the directories where they live.

Name That Directory

Directories became a standard DOS feature starting with Version 2.0, which accompanied the first IBM PC/XT machines equipped with hard drives. Before DOS 2.0, all the files on a disk—a floppy disk, that is—were housed in a single directory.

When multiple directories became possible, the initial directory became known as the home directory, or root directory. Additional subdirectories, which in turn could have nested subdirectories of their own, all branched from this central directory. The convention developed for tracing a path down from the root directory through one or more levels of subdirectories uses backslashes to separate the various subdirectory levels. The listing of the subdirectory names (such as \WORDPROC\LETTERS\MARCH\BUSINESS) is referred to as the path or pathname.

Directories enable you to put a large collection of files in a convenient order. You can isolate all the files associated with a given program or all the data files for a given project in a directory dedicated to those items. Some programs require files to be placed in directories with specific names; the installation process of such a program might create some of the directories it needs. You can create other directories yourself using DOS's MKDIR command (also called MD).

You can remove a directory with the RMDIR command (RD). You can't, however, easily change the name of a directory to something else with DOS commands, so select your directory names carefully.

Directory names follow the same rules as those for filenames. The name of each level of a directory may contain as many as eight characters and a three-character directory extension. You'll rarely need to use extensions on directory names, however, since you usually set up far fewer directories than files.

When you create a branching directory structure, you need to learn how to find your way around in that structure and tell your programs how to find the directory containing the files they need to use.

Finding Directory Names

A good first step to understanding directories is to keep track of the current directory and the current drive. You can find out what the current directory is by entering the CHDIR (CD) command at the DOS prompt. If you enter the command by itself, DOS tells you where you are by displaying the name of the current directory of the current drive.

But it's easy to have your system prompt always indicate the name of your current directory, along with the letter name of the current drive. If your present prompt provides only the drive name (a terse message such as C>), you can request a more informative prompt by entering this DOS command:

```
PROMPT $P $G
```

If your current directory is the subdirectory called PRINTER in the directory called UTILITY, DOS displays this message:

```
C:\UTILITY\PRINTER >
```

The TREE command is another tool for understanding the overall structure of your system's directories. When you invoke TREE, it displays the names of all levels of directories on a drive. Starting with DOS 4.0, the TREE command became much more useful; the list of directories it displays is arranged like a tree, clearly showing the relationships between directories. In DOS 5.0, the TREE command also will list the names of all files in each directory on a drive when you enter this command:

```
TREE \ /F
```

The backslash indicates that the report should start with the root directory.

If you have an earlier version of DOS with a less effective TREE command, you can get a similar full report of directory names and filenames by using the CHKDSK command with its /V parameter:

```
CHKDSK C: /V
```

The resulting display shows the name of every file on a drive and includes the file's full pathname. Think of the V in this parameter as standing for *verbose*, since the report it produces might be rather long.

Moving Among Directories

The CD command not only tells you more than your current location, it has the power to transport you to another directory. For example, if you're in a directory called \RECIPES, you can move to one of its subdirectories called \SOUPS by entering the following command:

```
CD SOUPS
```

If you start from the root directory, you could move to the \RECIPES\SOUPS directory one step at a time, as with these commands:

```
CD \RECIPES
CD SOUPS
```

Or, you could move there by issuing a single command:

```
CD \RECIPES\SOUPS
```

You can move back up through a series of directories one step at a time by using double periods (..), shorthand for the directory immediately above the current directory (often referred to as the parent directory). For instance, you can move back up from \RECIPES\SOUPS to \RECIPES with this command:

```
CD ..
```

The PATH command sets up a list of directories that DOS will search when a program can't be found in the current directory.

No matter which directory you are in and how deeply that directory is nested, you can move back to the drive's home directory by entering the following command at the DOS prompt:

```
CD \
```

Always keep in mind, though, that the CD command moves you only between directories on a single drive. If you want to switch from a directory on one drive to a directory on another of your drives, you need to enter a command for each of those procedures. For example, if you want to get from a directory on another drive to the \RECIPES\SOUPS\COLD directory that resides on drive C, you must enter these two commands:

```
C:
CD \RECIPES\SOUPS\COLD
```

Searching Directories Automatically

Now you know how to move from one directory to another as freely as a sorcerer can move from his tower to his cave. Wouldn't it also be nice to share some of this power with your servants—the application programs on your computer? DOS's PATH statement can command your programs to run—regardless of the directory you're in whenever you summon them.

If you haven't given DOS a path to follow, you might need to find your way to the directory where a program's files reside, before DOS can execute the program. For example, if you were logged onto drive A and wanted to run a utility program called CHANGO.EXE in the \UTILITY\FILTERS directory located on drive C, you might need to enter the following series of commands:

```
C:
CD \UTILITY\FILTERS
CHANGO
```

If you have DOS 3.0 or higher, you can run CHANGO by entering a single command line that gives DOS precise instructions on which directory to look in to find the program file:

```
C:\UTILITY\FILTERS\CHANGO
```

But a better way to do things is to use the PATH command to, once and for all, tell DOS where it can find CHANGO.EXE and all the other filter programs in that directory. Once you do that, you can simply enter this command:

```
CHANGO
```

regardless of the directory that is current when you enter the command.

The PATH command establishes a search path, a list of directories that DOS will search through when a program file can't be found in the current directory of the current drive. To see if a search path already exists, enter the PATH command by itself:

```
PATH
```

You may see a one-line message such as the following:

```
PATH C:\DOS;C:\;C:\UTILITY;C:\BATCH;
  D:\WORD;C:\GAMES\CARDS
```

In this example, the existing search path specifies six different directories that DOS will search through to find program files that have the .EXE and .COM extensions and batch program files (files containing a series of command lines) that have the .BAT extension. On this computer, any time you ask to run a program that DOS can't find in your current directory, DOS will search through the six directories listed in PATH to find the program file.

DOS will search the directories in the order they're listed in the path. If several program files with the same name are located in different directories, DOS will run the version it finds in the earliest directory named on the search path.

Note that even though the search path mentioned here already includes the \UTILITY directory, it won't help you find the CHANGO file until you add the \UTILITY\FILTERS directory to the search path. To add another directory to your system's search path, you need to enter a PATH command that names all the existing directories on the search path, as well as the new items. If, for instance, you were to enter only the one new item in a PATH command like this:

```
PATH C;\UTILITY\FILTERS
```

the original search path with six directories would be eliminated and replaced by a search path containing only one directory. You'd then have problems running even most DOS commands, which require files stored in this computer's C:\DOS directory.

Another DOS command, APPEND, is quite similar to the PATH command. It permits you to name additional directories that DOS will search when looking for files. APPEND can help you reach data files from the directories on its list, and it also can help you locate the .COM, .EXE, and .BAT program files that are the only types of files PATH can reach. One of APPEND's advantages is that you can use it to reach program files in additional directories without changing your PATH command's standard search path.

The APPEND command, introduced in DOS 3.2, has been improved in subsequent versions of DOS. If you're interested in using this command, you should first check the instructions for the version of DOS installed on your system. The APPEND command, if used carelessly, can lead you to open an unwanted data file (such as a document). Or you might even end up losing track of the directory where an edited version of a file is being saved.

Now that you can use some DOS commands to discover the names of files and directories and reach them rapidly, you can use other commands and programs with more powerful results. ■

James Langdell is a publications manager for Sun Microsystems and author of the PC Resource MS-DOS Insert Collection.

Max Ribble also contributed to the questions and answers on files and directories.

Questions & Answers

FILES AND DIRECTORIES

Q Copy doesn't seem to do everything I'd like it to do. How do I copy an entire directory and everything in it?

A Use the XCOPY command with its /S option so that all subdirectories below the directory you copy from—and the files those subdirectories contain—will be copied. For example, to copy the entire directory structure below C:\UTIL onto a disk in drive A, enter this command:

```
XCOPY C:\UTIL A: /S
```

This command creates any subdirectories that don't already exist on the destination drive. Keep in mind that you can't use this command to copy the directory structure of the current directory into a subdirectory of the current directory.

Q Do you recommend that I use the COPY or XCOPY command to move a file from one directory to another?

A You can use COPY or XCOPY to place a copy of a file in the desired directory. If you don't want the file to remain in its present directory location, you must remove it with the ERASE or DEL command. DOS doesn't have a command that will carry out both steps with a single command. However, a number of third-party utilities can do this. Many of them are named MOVE or MV, after the Unix command that can change the location and name of a file. Harry Bee's MOVE.BAT batch file provides a simple way to get the job done (see "Take Control with Batch Files" on page 55).

Q I want to shorten my directory names. How do I change the name of an existing directory?

A If you're asking, you probably discovered that a command such as:

```
REN THISDIR THATDIR
```

doesn't work. The DOS shell program provided with DOS 4.0 and later lets you rename directories. However, the only way to use DOS commands to change the name of a directory is to create a new directory with the desired name, copy the contents of the old directory into the new directory, remove all of the old files and directories that were under the unwanted name, and remove the old directory. Some third-party utilities (such as Unix look-alike commands named MV or MOVE) are available to rename directories in one step .

Q I've run into trouble cleaning up my hard drive. Why won't RMDIR always make a directory go away?

A RMDIR, which you can shorten to RD, will remove a directory only if it is empty. If you're having trouble removing a directory it may be because it contains files or subdirectories (even empty ones). RD also refuses to work if you try to remove a directory while you are in it—that is, if the directory you're trying to remove is the current directory of one of your drives.

The files may include ordinary program or data files (such as NORTON.EXE), hidden files (such as IBMBIO.COM, which typically resides in your root directory), or subdirectories that might contain files of their own. Names of ordinary files and subdirectories show up when you type DIR; to see hidden files and subdirectories, you need to call on DOSSHELL or the ATTRIB command. Even if a file is visible, you might not be able to delete it while its read-only attribute is set. To remove this restriction, enter the command:

```
ATTRIB -R filename
```

where *filename* is the name of the read-only file. After that, you can use DEL or ERASE to remove the file.

Q What are the period (.) and double period (..) characters that appear when I type DIR?

A These symbols are shorthand designators for the current working directory and the one above it (the so-called parent directory) in a disk's tree structure. Notice that the directory listing for a drive's root directory doesn't display the double period symbol. That's because there is no directory level above the root directory.

You can use these symbols in the commands you write. For instance, the command:

```
CD ..
```

will take you up one directory level (i.e., from C:\DOS\PATCHED to C:\DOS).

A single period is handy when you want DOS to pay attention only to a file it can find in the current directory, ignoring a file with an identical name that it might find in another directory mentioned in your PATH command. For example, if you think you have a RESTORE command file in your current directory but don't want to risk running any other version of RESTORE that the search path might find (BACKUP and RESTORE vary greatly among different DOS versions), you can enter the command:

```
.\ RESTORE
```

to assure that the command isn't run from anywhere other than the current directory.

Q Why does it take so long for DOS to find the programs listed in my search path?

A Every time you execute a command that isn't in the command processor (DIR and DEL are among the commands included in the command processor FORMAT and XCOPY are not) or in your current directory, DOS starts searching for a program file with that name by looking in each of the directories listed in the PATH command. The directories are searched in the order they are listed in the PATH command line.

To improve the overall speed of finding program files, place the names of directories containing programs and batch files that you want to start running quickly near the start of the PATH command in your AUTOEXEC.BAT file. You can afford to place directories containing larger application programs toward the end of the list, since the time it takes DOS to find these program files isn't significant compared to the time it will take for the programs themselves to start running. But fast response time is more desirable for utilities you use frequently or for those that DOS might require more time to find than the programs need to run.

Q I am new to DOS and subdirectories. How can I get quickly from one directory to another?

A If you're a fast touch typist, you may be happy with moving around by entering command lines such as:

```
C:\PROG\VIRUS>d:
```

or:

```
D:\HACK\DETHWORM>cd \telcom\uploads
```

However, you may find it preferable to reach a directory you use frequently by writing a batch file that takes you there from any current directory and drive. For example, to get quickly to the \TELCOM\UPLOADS directory on drive C, create a batch file named UPLOADS.BAT that contains these command lines:

```
D:
CD \TELCOM\UPLOADS
```

After that, you can get to that directory simply by typing the batch file name, UPLOADS. (For further examples of this technique, see "Take Control with Batch Files" on page 55.)

For fastest access, be sure you store such batch files in a directory that's named near the start of your search path.

Q How do I stop the directory listing from scrolling up past my screen before I can read it?

A DOS provides at least three ways to do this. You can display a more compact listing by asking DOS to display just the names of the files and directories; do this with the DIR /W command. You also can display full lines of information about each item with the DIR /P command, which displays one screen (or page) of directory items and then pauses until you ask to see the next screen's worth of listings.

Also, if you want to stop and view the listing at a specific point, you can press keys that will stop and start the flow of information to the screen. To stop the rush of data, press Ctrl-S. To resume the flow, press Ctrl-Q.

By the way, these two keyboard commands are a standard way to start and stop text displays on many computer systems, not just computers running DOS. This standard is known as X-ON/X-OFF flow control. ■

Tip 1

The Environmental PC

DOS's special message pad area, the environment space, provides an excellent means of customizing your computing sessions. The trouble is, once you begin putting the environment to work, it usually isn't long before you encounter an "Out of environment space" error message. If this happens to you, try adding the following line to your CONFIG.SYS file:

```
SHELL=C:\COMMAND.COM C:\ /P/E:1024
```

This command increases the size of your DOS environment from its default size, which varies according to the version of DOS you use, to 1024 bytes (1K). That's more than enough space to handle most situations.

DOS Goes Graphical

Novices and experienced users can benefit from getting to know Version 5.0's DOSSHELL graphical operating environment.

by Jack Nimersheim

Don't look now, but the world's gone GUI. GUI, pronounced like the word *gooey*, is an acronym for graphical user interface, the latest craze to sweep the computer industry. The Macintosh, advertised as "the machine for the rest of us" when it was introduced in the mid-1980s, was the first successful PC to sport a graphics-based operating environment. MS-DOS users got their initial taste of a true GUI in June 1990, when Microsoft Corp. finally shipped its much-touted and oft-delayed Windows 3.0.

One year later, in July 1991, DOS itself jumped over to the GUI side of the PC fence. That's when Microsoft released MS-DOS 5.0, the first major upgrade to its premier operating environment in over two years. One of DOS 5.0's most attractive new features is its DOSSHELL program, a graphics-based menuing system that offers an alternative to the familiar, yet frustrating, DOS system prompt. (Yes, I realize DOS 4.x also included something called DOSSHELL. The less said about this earlier, largely disappointing attempt to endow DOS with GUI capabilities, however, the better.)

Graphic Differences

The appeal of a GUI is undeniable. Standard DOS forces you to memorize and regurgitate long and often complex commands to accomplish even straightforward tasks. A GUI such as DOS 5.0's shell, on the other hand, relies on pull-down menus and dialog boxes to simplify many file- and disk-related operations.

Consider, for example, the following command:

```
COPY C:\WS\4\DOSSHELL.WS A:
```

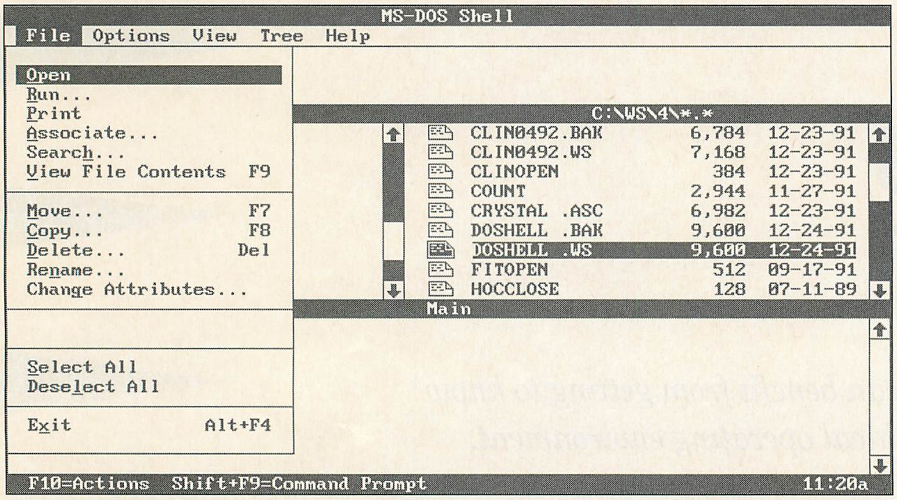
This is the standard DOS command used to copy to a floppy disk in drive A a file called DOSSHELL.WS that resides in a subdirectory on drive C named \WS\4.

Photos 1 and 2 illustrate how much simpler this action is to accomplish from within the DOS shell. Selecting the Copy option from the pull-down file menu with DOSSHELL.WS highlighted, as it is in Photo 1, displays the dialog box shown in Photo 2. To complete the copy operation, you can simply type A: at the underline prompt and press Enter. DOSSHELL handles everything else. Its interactive design, as demonstrated in this admittedly simple example, makes DOSSHELL especially attractive to someone just learning the ins and outs of DOS.

Something for Everyone

Experienced users also will discover several benefits, should they decide to abandon the familiar system prompt in favor of working within DOSSHELL. Chief among these is task switching, the ability to easily move between multiple application programs.

Don't misunderstand, however: DOSSHELL is not a multitasking envi-



```
C:\WS\4\*.*
CLIN0492.BAK     6,784  12-23-91
CLIN0492.WS      7,168  12-23-91
CLINOPEN           384  12-23-91
COUNT            2,944  11-27-91
CRYSTAL  .ASC    6,982  12-23-91
DOSHELL  .BAK    9,600  12-24-91
DOSHELL  .WS     9,600  12-24-91
FITOPEN            512  09-17-91
HOCCLOSE           128  07-11-89
```

Photo 1. To copy a file within DOSSHELL, first highlight the file you wish to copy, then select the Copy option from the File menu.

ronment. Unlike true multitasking systems—Windows and Quarterdeck Office System's Desqview, to name but two—DOSSHELL doesn't make your system resources (CPU, memory, printer, and the like) available to multiple applications concurrently. Rather, as the program's name implies, DOSSHELL's task switcher lets you jump quickly among different applications, temporarily suspending the operations of programs not in use. (To fall back on a familiar analogy, task switching resembles listening to a radio. Although, in theory, all stations are always available, you can hear only the one to which you are currently tuned.)

Enabling Task Switching

Before you can use the DOS shell's task switcher, it must be turned on. To do this using input from the keyboard:

- switch to the view called Program/File Lists by pressing Alt-V and then *F* (unless, of course, that view already is displayed)
- press Alt-O to display the pull-down Options menu
- press *E* to select Enable Task Swapper

After task switching is enabled, the Active Task List will appear in the lower righthand corner of the display. When you first turn on task switching, the Active Task List is empty because you haven't started any applications from within DOSSHELL. You can add an application to the task list in several ways. One involves selecting the Run option in DOSSHELL's File menu. If, for example, you want to start the MS-DOS 5.0 text editor:

- press Alt-F to display DOSSHELL's File menu
- press *R* to select Run
- when DOSSHELL displays its Run dialog box, type:

```
EDIT
```

the command normally used to start this application

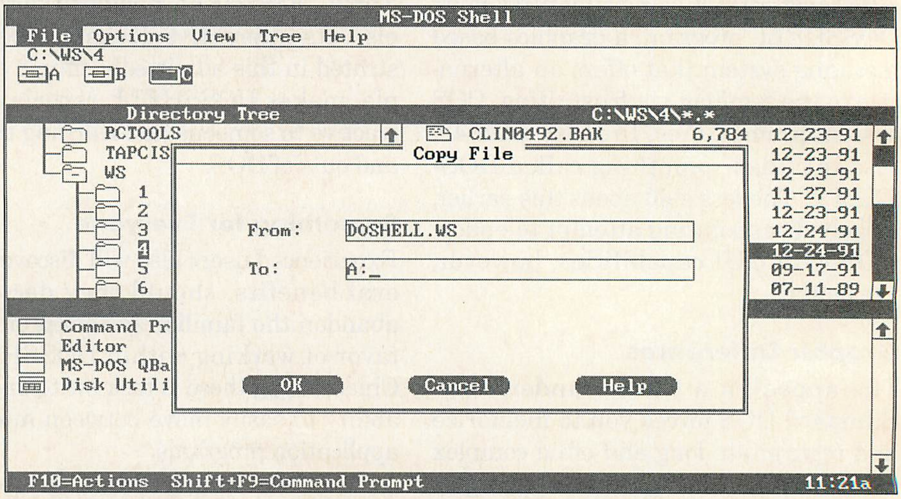
- press Enter

After a few seconds, you'll see a message welcoming you to the MS-DOS Editor. Pressing Esc removes this message screen and brings up the editor. Now, try an experiment by pressing Alt-Tab. You'll notice that the Active Task List contains an item called EDIT, the command you used to run the DOS 5.0 text editor. You use the Alt-Tab key combination to switch between multiple applications running in a task-switching session. Normally, when you press Alt-Tab, it cycles you through any open applications in "round-robin" fashion, but because EDIT is the only program currently running, pressing Alt-Tab returned you to the main DOSSHELL display.

Working with Program Groups

Another useful feature of DOSSHELL is its ability to manage what Microsoft calls program groups. During installation, DOS creates a Main program group and places four items in it: Command Prompt, Editor, MS-DOS QBasic, and Disk Utilities. To take full advantage of this feature, you'll need to learn how to add new items representing your favorite applications to the Main program group.

Stated simply, a program group is a group of related applications and data files that is easily accessible from the DOS shell. You might, for example, use this feature to organize



From: DOSHELL.WS

To: A:_

Photo 2. The Copy File dialog box lets you enter the destination for the file you wish to copy.

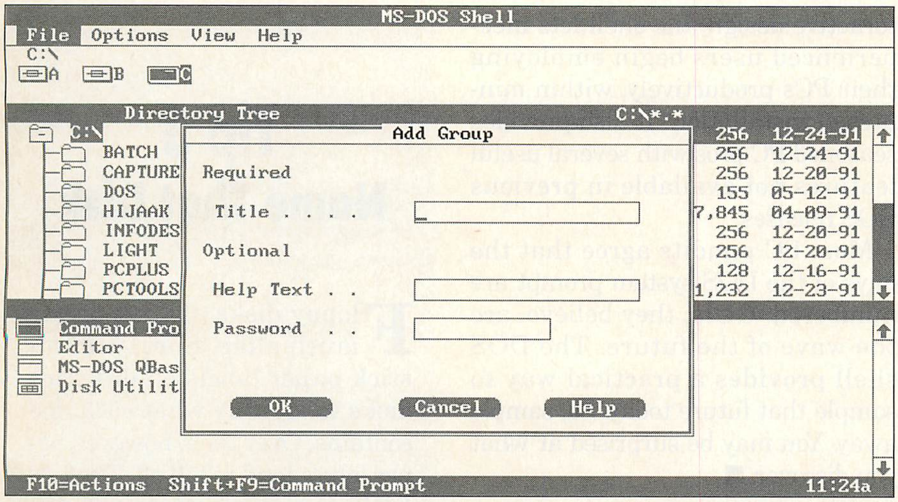


Photo 3. DOSSHELL makes it easy to create your own program group. Simply highlight the Main group window, then select New from the File menu.

several DOS commands or other utility programs you use regularly into a program group called Misc. Utilities. The first step in the process is creating a new program group to hold these utilities:

- press the Tab key until the highlight bar is in the Main program group window (within the Program/File Lists view)
- press Alt-F to display the File menu
- press *N* to select the New option
- press the up-arrow key to select the Program Group option
- press Enter to choose OK
- When DOSSHELL displays the Add Group dialog box (see Photo 3), type the title field of the box—Misc. Utilities.

The two other fields shown in this dialog box, Help Text and Password, are optional. Use the Help Text field if you want to enter a help message that the DOS shell will display whenever you access on-line help for this particular program group. A help message can contain as many as 256 characters. To protect a program group against unauthorized access, use the Password field to specify a password for the group. Leaving the password field blank allows anyone who has access to your PC to use this program group.

After you press Enter to tell DOS you're ready to add the new program group, MS-DOS adds the new group to the Main window and returns you to the DOS shell display. Once a program group exists, you can use the following steps to add items to it:

- if necessary, use the arrow keys to highlight Misc. Utilities
- press Enter to activate this program group
- press Alt-F to display the File menu
- press *N* to select the New option
- press Enter to tell DOS you want to add a new program item

DOSSHELL then displays the Add Program dialog box, which you use to specify the information DOS will need to run this item whenever you select it from your Misc. Utilities group. In the following exercise, you'll create a program item that will report on the current status of your PC's memory:

- type Check Memory in the Program Title field
- press Tab
- type MEM.EXE in the Commands field
- press Tab again, and then type the name of the directory on your hard disk in which you store your MS-DOS 5.0 files
- press Tab one more time

This advances the cursor to the Application Shortcut Key field. This allows you to specify a key combination that can be used to quickly run your new program item. If you wish, you can, for example, enter Shift-M in this field. Doing so will cause DOSSHELL to run the Check Memory routine whenever you hold down the Shift key and press *M* while working in the shell.

- Press Enter to tell DOS you're ready to add the new program item

Your Misc. Utilities program group should now contain two items: Main and Check Memory. Main is an empty entry that provides a link to the programs one level up in your directory structure. Selecting Main and pressing Enter returns you to the Main program group.

Let's see what happens when you select the second item, Check Memory. To perform a memory check:

- highlight the Check Memory option by pressing the up-arrow key if this choice isn't highlighted already
- press Enter to select the item and run MEM.EXE, a DOS utility program that analyzes your PC's memory and returns information on the memory available. Notice that DOS temporarily suspends execution after running MEM.EXE and displays a "Press any key to return to MS-DOS Shell" message. This happens whenever the "Pause after exit" option in the Add Program dialog box is marked as active. If you turn off this option, DOS will return to the shell display as soon as a program finishes executing. You'll need to make sure the Pause option is activated, therefore, whenever a program item displays information you want to read before returning to DOSSHELL. To return to the DOSSHELL display after reading the results of your memory analysis, press any key.

Setting up the appropriate program groups lets you design a DOS shell environment that complements your work habits. If your job entails financial planning, for example, you could create a program group, called BUDGETS, consisting of a spreadsheet program such as Lotus 1-2-3 and the individual data files for each of your major projects.

Impressive, but Not Perfect

One major advantage DOSSHELL possesses over other GUIs is that it will run on any IBM-compatible PC. Unlike Windows, which requires at least an 80286 computer with a minimum of 2MB of RAM and an EGA display, DOSSHELL works perfectly well on an 8088- or 8086-based PC with a minimum of 640K of RAM. You even can use DOSSHELL with a nongraphics monitor, although its interface is decidedly less graphical when DOSSHELL is configured to run in text mode.

As the examples in this article demonstrate, DOSSHELL responds well to keyboard commands. Adding a mouse to your system simplifies matters further. For instance, a mouse transforms the procedures for carrying out a COPY command into a one-step procedure. You merely point to the file you want to copy, hold down the Ctrl key and the left mouse button, then drag the mouse pointer to the new directory. Releasing the mouse button completes the COPY operation. What could be easier?

As impressive as it is, however, DOSSHELL does have some obvious shortcomings. It would be nice, for example, if you could cut and paste information between different applications running in the Active Task List. You can't.

Another useful feature would be an option to allow DOSSHELL to swap programs to RAM during task switching—provided your computer possessed enough memory to support this technique. Currently, all task swapping utilizes temporary disk files, an approach that's considerably slower than would be possible if DOSSHELL shuffled applications in and out of RAM.

A Good Beginning

So, is DOS 5.0's DOSSHELL worth using? In a word: Yes. Given its interactive design, the shell lets inexperienced users begin employing their PCs productively, within minutes of installation. It also provides seasoned PC pros with several useful features not available in previous DOS releases.

Most PC pundits agree that the days of the DOS system prompt are numbered. GUIs, they believe, are the wave of the future. The DOS shell provides a practical way to sample that future today. So, sample away. You may be surprised at what you discover. ■

Computer columnist and free-lance writer Jack Nimersheim has been using and writing about PCs for over 10 years. Some of the material in this article was excerpted from his best-selling 10 Minute Guide to MS-DOS 5.0 *(SAMS, May 1991), which currently is in its fifth printing.*

Tip 2

A Helping Hand

Beginning with DOS 5.0, the most ambiguous PC operating system got a little less cryptic. A new feature lets you access on-line help for specific commands, whenever you're working at the DOS prompt. Requesting help is a simple procedure. All you do is enter the command for which you want information, followed by /?. For example, the following command displays information about the various ways in which you can enter the DIR command:

```
DIR /?
```

Tip 3

Name That Disk

Floppy disks all look pretty much alike. Sure, you can stick paper labels on different disks to identify what each one contains. Over time, however, paper labels tend to fall off. The LABEL command offers a much more permanent identification method—it lets you create a label and save it as a hidden file on a disk. That label is displayed each time you request a directory listing of the disk's contents.

You can use the following command, for example, to create a label for a floppy disk in drive A:

```
LABEL A:
```

DOS responds by displaying the label currently assigned to the floppy disk in drive A, if one exists, and then displays the following prompt:

```
Volume label (11 characters,
ENTER for none)?
```

At this point, you merely type in the label name you want assigned to this particular disk and press the Enter key.

The LABEL command was introduced in MS-DOS 3.1. If you have an older version of DOS, you may label a disk only during formatting. To accomplish this, you must add the /V switch (for *volume*) to the FORMAT command, as follows:

```
FORMAT A: /V
```

After formatting a floppy disk, older versions of DOS prompt you to assign it a volume label.

Managing DOS Setup Files

Understanding CONFIG.SYS and AUTOEXEC.BAT—and preparing for inevitable bootup problems—can save you time and frustration.

by Hardin Brothers

My worst computing nightmare does not involve a hard disk crash, a power failure during a critical disk operation, or a lightning strike. Good backups and some common sense can protect me against those possibilities. But I'm 99 percent positive that the installation routine for a new application will one day make my computer unusable for no apparent reason.

There was a time when the first pages of all software documentation had instructions that read something like, "Copy the files on the distribution disk to a subdirectory on your hard disk. If you want to access the program from other directories, add the name of its subdirectory to your AUTOEXEC.BAT file's PATH statement. This program requires that you specify at least 20 files in your CONFIG.SYS file. We recommend a minimum of 10 buffers, as well." The manual would then explain each of these operations so that novice users could understand how to carry them out.

But today, installation instructions are usually limited to, "Put disk 1 in drive A and type A:INSTALL." The installation program takes over, copies (and often uncompresses) the distribution files. Then the installation program makes changes to your AUTOEXEC.BAT and CONFIG.SYS files for you. Some programs then automatically reboot your computer so that the changes take effect.

Great! For novice users and computers that run only two or three applications, these automatic installation programs are a dream. But if you have complex and carefully crafted configuration files, if your PATH statement already is pushing the DOS length limit, or if you run hardware or software that the installation program doesn't know about, your computer may lock up completely as it reboots.

This isn't a simple lockup that you can solve by swearing a couple of times and pressing Ctrl-Alt-Del. It's a full-fledged, I-won't-run-anymore-no-matter-what-you-do lockup that can prompt even the calmest computer user to shred the program's documentation into confetti and launch a blistering verbal attack on the product's technical support staff.

The most frustrating thing about this danger is that no one talks about it or even seems to acknowledge that it exists. But once you understand why it might happen, you can protect yourself with less effort than it takes to back up last week's data.

Flexibility Brings Responsibility

If DOS were a rigid, monolithic system, new applications would present little danger. But DOS is much more flexible than most people realize. Its two setup files, CONFIG.SYS and AUTOEXEC.BAT, let you set up DOS to match almost any work style and any combination of applications and utility programs.

When you boot up your computer, the machine's built-in BIOS routines load an operating system (usually DOS) from the floppy disk in drive A. If that drive doesn't have a disk, the BIOS next looks to the boot partition of your hard disk. Whether the BIOS finds what it's looking for on a floppy disk or a hard disk, its next step is to find a special bootstrap loader program on the disk, which in turn loads DOS's core functions, the so-called DOS kernel. While it loads and initializes itself, DOS reads your CONFIG.SYS file so that it can set itself up to suit your needs. As soon as the installation of the operating system is complete, DOS reads and executes each of the instructions in your AUTOEXEC.BAT file.

Generally, you don't have an opportunity to interrupt this process, although you occasionally can halt AUTOEXEC.BAT by pressing Ctrl-Break. But if your CONFIG.SYS or AUTOEXEC.BAT file contains an error, DOS and your computer may not be able to boot up at all, or your computer may become unaccountably flaky, doing unexpected things at almost any time.

The cure for these ills is easy: Simply remove or correct the offending commands in CONFIG.SYS and AUTOEXEC.BAT. But if you can't make your computer operate reliably—or even operate at all—how can you edit these two files? That's the catch that can exasperate even a computer wizard.

Avoid the Panic

You can avoid problems of this kind by taking several precautions. The first is to view all automatic installation programs as potential troublemakers. If your system and configuration files are essentially the same as the ones that the author of the program used, everything should be fine. But if your system is at all unusual—that is, if it includes *any* provisions for hardware or software unfamiliar to the developers who

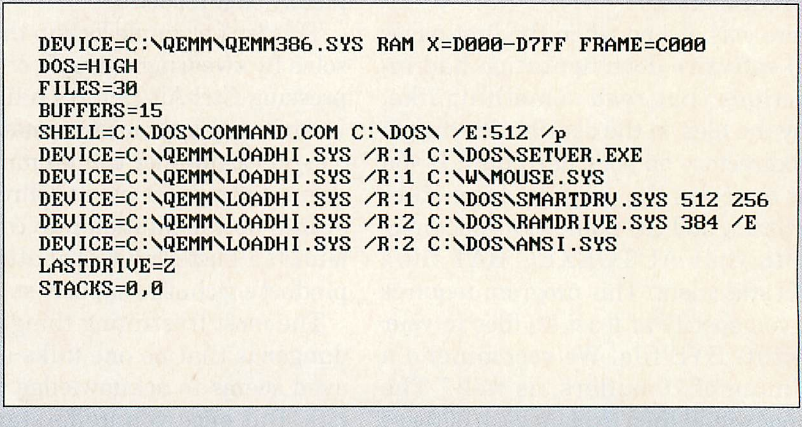
THE ANATOMY OF CONFIGURATION FILES

The content of your CONFIG.SYS and AUTOEXEC.BAT files can be as unique as fingerprints. The best configuration for your system depends on your hardware, the kinds of applications you use, and your work habits. As your system and its software change over time, so may your configuration files.

Below is an inside look at my CONFIG.SYS and AUTOEXEC.BAT files. To put them in perspective, my system is a 386SX machine with 3MB of extended memory and a 40MB hard drive. I use MS-DOS 5.0. (If you have an earlier version of DOS, I highly recommend the upgrade to Version 5.0.) I spend a good deal of my day running Microsoft Windows 3.0 and use mostly Windows applications, with a few non-Windows programs in the mix. My machine is connected to a small local-area network, giving me access to additional disk storage and a laser printer.

Let me emphasize that my configuration files are not at all likely to be right for your machine. Do not try to change your files to match mine. Rather, as you examine my configuration, look for ideas that you can incorporate into your own setup to make your system more efficient and easier to use.

A Look at CONFIG.SYS



```
DEVICE=C:\QEMM\QEMM386.SYS RAM X=D000-D7FF FRAME=C000
DOS=HIGH
FILES=30
BUFFERS=15
SHELL=C:\DOS\COMMAND.COM C:\DOS\ /E:512 /p
DEVICE=C:\QEMM\LOADHI.SYS /R:1 C:\DOS\SETVER.EXE
DEVICE=C:\QEMM\LOADHI.SYS /R:1 C:\W\MOUSE.SYS
DEVICE=C:\QEMM\LOADHI.SYS /R:1 C:\DOS\SMARTDRV.SYS 512 256
DEVICE=C:\QEMM\LOADHI.SYS /R:2 C:\DOS\RAMDRIVE.SYS 384 /E
DEVICE=C:\QEMM\LOADHI.SYS /R:2 C:\DOS\ANSI.SYS
LASTDRIVE=Z
STACKS=0,0
```

CONFIG.SYS

DEVICE=C:\QEMM\QEMM386.SYS RAM X=D000-D7FF FRAME=C000

This first line installs the QEMM386 memory manager from Quarterdeck Office Systems. QEMM386 takes the place of both DOS 5.0 memory managers, HIMEM.SYS and EMM386. QEMM386 is easier to use and more flexible. The RAM switch on this line tells QEMM386 to make high memory available for loading device drivers and terminate-and-stay-resident programs, so-called TSRs.

wrote the automatic installation program—you may face problems.

For added security, always have a bootable floppy disk ready whenever you install a new program. To create a bootable disk, put a new, high-quality floppy disk in drive A, and type:

```
FORMAT A: /S
```

This formats the disk and installs the hidden DOS system files on it. Some versions of FORMAT /S also copy COMMAND.COM to the new floppy disk. If your version of FORMAT doesn't do this, copy COMMAND.COM to the boot floppy after it is formatted. If you use Microsoft Windows 3.0, you also might want to keep a backup copy of WIN.INI on your emergency disk for future reference.

Next, copy your current CONFIG.SYS and AUTOEXEC.BAT files to the floppy disk, but change their file extensions so they won't be invoked automatically if you have to boot your system from the floppy disk. If you need certain CONFIG.SYS commands to make your system work at all—to run disk partitioning software, for example—create a new, streamlined configuration file that contains only the most essential commands and save that file on your bootable floppy disk as CONFIG.SYS.

The most important step, however, is to copy a simple file editor to your emergency bootable floppy disk. That way, if an installation program somehow messes up the configuration files on your hard disk, you can use this editor to make corrections. The DOS Edlin program or DOS 5.0's MS-DOS Editor is fine if you don't have another basic editor.

Before you stash the emergency boot floppy away, be sure that it really works by putting it through a few simple tests. Put it in drive A and reboot your computer. Make sure that the computer boots up from the floppy disk and that you can use the text

The X=D000-D7FF switch tells QEMM386 not to manage the memory in that address range. My computer is connected to a network, and the network interface card uses those addresses. FRAME=C000 tells QEMM386 to put the expanded memory page frame at a particular address, again to avoid conflict with my network card. Because I want to take advantage of the capabilities of QEMM386 later in my CONFIG.SYS file, I always install QEMM386 in the very first line of CONFIG.SYS.

DOS=HIGH

This particular DOS 5.0 command tells DOS to load a portion of itself into the high memory area (HMA), the first 64K of extended memory. Loading DOS into this area leaves more of your system's precious 640K of conventional memory available for running application programs. You must install a memory manager such as QEMM386 or HIMEM.SYS before you can use the DOS=HIGH command to load part of DOS into the HMA.

FILES=30
BUFFERS=15

The FILES command determines the number of files DOS can access at one time. If you don't specify a number of files, the default is 8. A more typical value is 20, but because I run Windows 3.0 on my machine, I've specified 30 files.

BUFFERS determines how much memory will be reserved to hold data during disk read and write operations. Values from 10 to 30 are typical. (See "Questions & Answers: Improving Productivity" on page 35 for a more detailed explanation of how BUFFERS relates to other CONFIG.SYS options.)

SHELL=C:\DOS\COMMAND.COM C:\DOS\ /E:512 /P

The SHELL command specifies the filename and location of your command interpreter. In most cases, this is COMMAND.COM, the command interpreter that comes with DOS, but other command interpreters are available.

If you omit the SHELL command from CONFIG.SYS, DOS looks for COMMAND.COM in the root directory of your start-up drive. An error will occur if it's not there. By using the SHELL command, you can keep COMMAND.COM in a directory of your choice.

The SHELL command doesn't accept any parameters or switches, but if you want to use parameters or switches with the command interpreter, you can add them to the end of the SHELL line. In this case, the C:\DOS\ parameter tells COMMAND.COM where to find itself when it needs to reload part of itself. The /E:512 switch increases the size of the DOS environment to 512 bytes from the default of 160 or 256 bytes (depending on your version of DOS). The DOS environment is an area of memory used to store the information in PATH, PROMPT, and SET statements. As your configuration becomes more complex, the default environment size may not be large enough. Increasing this setting to 512 bytes gives you plenty of room to grow. Finally, the /P switch makes the command interpreter permanent. If you fail to specify this switch in your SHELL command, DOS won't run your AUTOEXEC.BAT file when you start up your computer.

editor on drive A to edit the configuration files on your hard disk. Once the disk has passed these tests, it's safe to put it in your desk drawer. Then, simply reboot your computer to invoke the configuration files on your hard disk.

Now you can run any sort of installation program, or make any changes to your CONFIG.SYS and AUTOEXEC.BAT files, without worrying. If something goes wrong, you can simply reboot with your emergency disk in drive A, edit the configuration files on your hard disk, remove the emergency disk, and reboot your computer normally. The emergency disk will cost you less than $5 in both time and money to create, yet it could keep you from having to reformat your hard disk to regain control of your system.

With copies of your configuration files on your emergency floppy, you also have something to compare against the setup files on your hard disk. That might prove invaluable in narrowing your search for errors that an automatic installation program may have caused.

The Dangerous Commands

Theoretically, an installation program can add or change any command in your configuration files. But in practice, most programs alter only a few commands.

In your CONFIG.SYS file, installation programs generally touch only three or four commands: FILES, BUFFERS, DEVICE, and INSTALL statements. (The last of these shows up only in DOS 4.0 and later.) A FILES statement sets the maximum number of files that can be open at any one time. If the number of files is too small, programs that need to work with several files at once—such as database managers—may not be able to run well or at all. Some applications can diagnose the source of the problem correctly; others may report that they can't open a file or can't find a file that you know exists on your system.

```
DEVICE=C:\QEMM\LOADHI.SYS /R:1 C:\DOS\SETVER.EXE
DEVICE=C:\QEMM\LOADHI.SYS /R:1 C:\W\MOUSE.SYS
DEVICE=C:\QEMM\LOADHI.SYS /R:1 C: \DOS\SMARTDRV.SYS 512 256
DEVICE=C:\QEMM\LOADHI.SYS /R:2 C:\DOS\RAMDRIVE.SYS 384 /E
DEVICE=C:\QEMM\LOADHI.SYS /R:2 C:\DOS\ANSI.SYS
```

The above five lines install device drivers, special-purpose programs that interact with the hardware in my system. Because I am using QEMM386 as my memory manager, I use its LOADHI facility to load these device drivers into upper memory blocks, thus saving as much of my 640K of conventional memory as possible for running applications. If you use HIMEM.SYS and EMM386.EXE, the memory managers that come with DOS, you often can accomplish the same thing by using the DEVICEHIGH and LOADHIGH commands. Loading device drivers into upper memory is possible only on 386 and 486 systems.

Here's a rundown on what the five lines accomplish:

- SETVER.EXE is available only in DOS 5.0. It sets the DOS version number that DOS 5.0 will report to an application. It's used when you have an application that fails to recognize DOS 5.0 as a valid DOS version.
- MOUSE.SYS is the driver required to enable my Microsoft mouse. This line simply makes it possible for me to use a mouse on my system.
- SMARTDRV.SYS is a disk-caching program that comes with Windows 3.0 and with DOS 5.0. A disk cache keeps some or all of a recently retrieved disk file in memory, on the assumption that you soon may need to read that file again. It can improve dramatically the perceived performance of a hard drive. The first parameter (512) is the normal size of the cache in kilobytes. The second parameter (256) is the size of the cache in kilobytes when Windows is started.
- The line containing the RAMDRIVE.SYS command creates a 384K virtual disk, or RAM disk. The /E switch puts the virtual disk in extended memory.

  A virtual disk is a portion of memory that appears to the system as an additional hard drive. It is assigned the next drive letter available on your system. Because I have two floppy drives (assigned as drives A and B) and one hard drive (assigned as drive C), RAMDRIVE.SYS creates the virtual disk as drive D. When I discuss my AUTOEXEC.BAT below, I'll explain how a virtual disk can speed up your system.
- ANSI.SYS is a device driver that gives you considerable control over your system's display and keyboard. You can use it to redefine keys and to change the screen colors and cursor behavior.

```
LASTDRIVE=Z
```

This line specifies the highest drive letter that my system will recognize. Although my computer contains only four drives (two floppies, a hard drive, and a virtual disk), I'm connected to a network that gives me access to many more drives, so I've set LASTDRIVE to let me access drives A through Z.

```
STACKS=0,0
```

This lines tells DOS not to allocate any memory for hardware interrupt processing. This saves a little memory and seldom causes a problem. If you encounter unexplained system lockups with STACKS=0,0, try removing the STACKS command to see if the problem goes away.

A Look at AUTOEXEC.BAT

```
PATH C:\DOS;..;C:\W;C:\W\WINBATCH;C:\QEMM;C:\BATCH;C:\UTILS;C:\NORTON
VERIFY ON
PROMPT $P -$G
C:\NORTON\FR C: /SAVE
C:\UTILS\RESETNL.COM
SET USERID=JDD
SET DSZPORT=1
SET XFERDIR=C:\XFER
SET TMP=D:\
SET TEMP=D:\
C:\QEMM\LOADHI /R:2 C:\DOS\DOSKEY.COM
CALL LAN
```

AUTOEXEC.BAT

PATH C:\DOS;..;C:\W;C:\W\WINBATCH;C:\QEMM;C:\BATCH; C:\UTILS; C:\NORTON

My PATH command contains only one unusual entry. The second item, right after C:\DOS, consists of just two periods (..). This is a bit of DOS shorthand and refers to the directory that is one level above the current directory, whatever the current directory may be.

Let's say my word processor is in a directory called C:\WORD and my documents are in the directory C:\WORD\DOCS. If I am at the DOS prompt in the C:\WORD\DOCS directory and want to run my word processor, I can type the command to start the word processor and, because I have (..) listed in my path, DOS will look one level above the current directory, find my word processor in C:\WORD, and run it. It may not get used very often, but putting the double period entry in your path can be a worthwhile trick.

I try to keep my directory names rather short. That's because DOS imposes a 127-character limit on the PATH command. As you add programs and directories to your hard drive, this limit can pose a problem, so it's important to plan ahead, even if 127 characters seems like a lot to you now. For instance, when I installed Windows 3.0 on my computer, I specified C:\W instead of C:\WINDOWS as the installation directory. In my PATH command, this change saves 18 characters because the C:\W directory is referenced three times.

VERIFY ON

This command instructs DOS to verify that files are written correctly to disk. It does not check to see if the file itself contains any errors nor will it check to ensure that a file being copied is exactly the same as the original. VERIFY merely assures that the file is physically written to the disk in the correct manner and that data isn't written to a bad disk sector. Placing a VERIFY ON statement in your AUTOEXEC.BAT file slows down disk writes somewhat but provides a simple check of data integrity.

A BUFFERS statement establishes the number of file buffers that DOS will maintain as a basic disk cache. If this number is too small, programs may run too slowly. If either the FILES or BUFFERS setting is too large, DOS will waste memory space, and you may not be able to run large applications that once worked fine.

Most installation programs know the minimum or optimum number of files and buffers that their programs need. If you already have FILES and BUFFERS set to large values, these programs will leave the settings alone. But if you have set smaller values, automatic installation programs will reset them. If your CONFIG.SYS doesn't happen to have a FILES or BUFFERS statement at all, the installation programs will add them to your CONFIG.SYS file.

Although technical differences exist between the DEVICE and INSTALL commands, both are used to load memory-resident programs into memory during DOS's setup procedures. Sometimes system failures are caused by terminate-and-stay-resident (TSR) programs being loaded in the wrong order. To make everything work, you may have to relocate one or more lines in your CONFIG.SYS file. Problems based on order occur most often if you use a special disk partitioning or compression program that needs to be loaded before the new program can be found, or if you use a memory manager that needs to be loaded first. Occasionally, an installation program may include the wrong pathname or the wrong options in a DEVICE or INSTALL command.

If an installation program adds a DEVICE or INSTALL statement to your CONFIG.SYS file, and you see an error message flash by on your screen during system initialization, boot up again and press the Pause key as soon as the message appears. If the message says that the file could not be found, edit your CONFIG.SYS file to include the correct path for the file. If other errors appear, you will have to examine the offending lines (DOS tells you which lines contain errors) in your CONFIG.SYS file to see if they should be in a different order or if you need to edit the options on some lines.

AUTOEXEC.BAT files also hold potential traps for unwary installation programs. These programs might, for instance, add a subdirectory to your PATH statement so that the new application can be run from any subdirectory on your system. However, if you have a long PATH setting or use multiple shorter PATH statements, an installation program may well trigger system errors by exceeding the 127-character limit. The various versions of DOS react to this problem in different ways, sometimes by locking up.

The solution is to edit the PATH statements in your AUTOEXEC.BAT file so that the total setting is never too long. Or you may want to write short batch files that establish a PATH setting and then invoke the appropriate application, instead of trying to have all applications available at all times.

Other potential sources of trouble are the installation routines for memory-resident utilities, which sometimes add a program's necessary start-up commands to the very end of your AUTOEXEC.BAT file. If you already start a program such as XTree, The Norton Commander, or Windows from your AUTOEXEC.BAT file, the new commands will never be executed, and you may think that the new program does not work properly. To set things right, simply move the lines that start your new program to an earlier location in the file and try again.

Understanding Your Configuration

Once you have a new program successfully installed and running, com-

PROMPT $P -$G

The default command line prompt in DOS is a rather uninformative one consisting of the drive letter and the greater-than sign (>). The PROMPT command allows you to create a much more informative and attractive prompt. Special codes that start with a dollar sign ($) are used to customize the prompt that you see displayed. The one I use displays the current drive and directory ($P) followed by an arrow consisting of a hyphen and the greater-than sign ($G). Many other prompts are possible.

Here's a PROMPT statement that displays the date and time on one line and the current directory on the line below:

```
PROMPT $D $T$_$P$G
```

This one greets me by name:

```
PROMPT Hello, Jeff!$_$P$G
```

Your DOS manual lists all of the dollar-sign codes permissible in a PROMPT command.

C:\NORTON\FR C: /SAVE

This line runs The Norton Utilities' Format Recover (FR) program each time I turn on or reboot my computer. Format Recover saves certain key information about my hard drive, its directories, and its files to a special area on the hard disk. If the drive ever is accidentally reformatted or if the drive's file-allocation table (FAT) ever becomes damaged, the information saved by Format Recover will greatly simplify the task of rebuilding the structure of the drive. I began running Format Recover not long after losing the entire contents of a 20MB hard drive when its FAT was damaged.

C:\UTILS\RESETNL.COM

RESETNL.COM is a small program that appeared in the first issue of *DOS Resource Guide*. Running it turns off the Num Lock key on your keyboard so that you can use the Insert, Delete, and arrow keys on the numeric keypad, instead of using it to produce numbers. It's in my AUTOEXEC.BAT file because I always want my Num Lock to be off when I start my computer.

SET USERID=JDD
SET DSZPORT=1
SET XFERDIR=C:\XFER

These three commands assign values to three DOS environment variables. These variables are used by some of the software I'm running on my machine. The first one, USERID, is required by my office's local-area network (LAN) software, and it is simply set to my initials. The other two are variables required by a communications program to indicate which serial communications port is to be used and the directory in which downloaded files should be stored.

SET commands such as these are usually specific to a particular application pare your operating copies of CONFIG.SYS and AUTOEXEC.BAT with the copies of those files residing on your emergency boot disk. If the files have changed, put comments in the new files with REM statements. Any line in either file that starts with a REM statement is a remark and, as such, is ignored by DOS (early versions of DOS report harmless errors when they encounter REM lines in a CONFIG.SYS file).

Each REM should tell you which program added or changed the line. You may also want to include the version number and date that you installed the program. If the program made a change to an existing line, you also should include the original line in the file's remarks. As soon as you add the remarks and are positive that the new files are in working order, copy them onto your bootup disk.

These remarks will help you understand your configuration files and the purpose of each line. If you later decide to remove the program, the remark statements will help you restore the configuration files properly. They also will provide a good troubleshooting guide if you ever have to find and fix an error in either file or if you need to discuss the files with a software company's technical support department.

As you add programs and remove others, your configuration files will become as personal as your fingerprints. If you keep them well-documented and always have an equally well-maintained bootup disk handy, you can make changes, add software, and configure your computer to your own liking. And you'll never again have to worry about having an inopportune lockup keep you from using your computer when you desperately need to get work done. ■

Free-lance writer Hardin Brothers has been working with and writing about computers for 13 years.

---

and often are inserted in your AUTOEXEC.BAT file by the program that installs the application. So don't be surprised to find new SET commands in your AUTOEXEC.BAT file after installing a new program.

SET TMP=D:\
SET TEMP=D:\

Software applications often create and use temporary files while they are active. Many programs look for the DOS environment variables TEMP and TMP to find out where you would like the temporary files to be stored. Because the speed with which temporary files are created and accessed can have a big effect on the speed of a program's operation, it's advantageous to set the TEMP and TMP variables to point to the fastest disk drive in your system.

In this example, the TEMP and TMP variables are set to point to the virtual disk (RAM disk) created in line nine of my CONFIG.SYS file. Because a virtual disk stores information in a portion of the computer's memory, it is much faster than any physical drive. One possible disadvantage is that the storage capacity of a virtual disk is usually rather small. In my system, I create a virtual disk that holds only 384K of data. If I start to use programs that create extremely large temporary files, 384K may not be enough to hold the files. Because I don't have enough memory to make the virtual disk much larger, the solution would be to set the TEMP and TMP variables to point to a directory on drive C, my hard drive. In that case, I would create a directory for temporary files on drive C. The two lines to do this might look like the following:

```
SET TMP=C:\TEMPSTUF
SET TEMP=C:\TEMPSTUF
```

Using a physical hard drive for storing temporary files is slower than using a virtual disk, but it's your only choice if memory limitations restrict the size of the virtual disk you can create. If your machine has 640K of conventional memory and no expanded or extended memory, for example, your only choice is to use your hard drive for temporary files.

One final reminder: Don't forget that any data stored in a virtual disk is lost when you turn your computer off or reboot. Don't store critical data files on a virtual disk unless you can guarantee that you'll remember to copy them to a physical drive before shutting down or rebooting.

C:\QEMM\LOADHI /R:2 C:\DOS\DOSKEY.COM

This line uses the LOADHI facility of the QEMM386 memory manager to load DOSKEY.COM into an upper memory block. DOSKEY is a DOS 5.0 program that lets you edit command lines, recall a series of DOS commands, and create macros.

CALL LAN

This line starts a batch file called LAN.BAT, which loads the local-area network software and connects my machine to the network. The CALL command lets you call a second batch program from within another without causing the first batch program to stop.

—Jeff DeTray

Questions & Answers

SETTING UP DOS

Q What is the best order for the commands in my CONFIG.SYS file?

A Many people put the FILES and BUFFERS commands at the beginning of CONFIG.SYS, DOS configuration commands such as LASTDRIVE in the middle, and DEVICE and INSTALL commands at the end. But this is only for convenience; it makes sense to group the commands so that humans can understand the organization of the file more easily. In most circumstances, the order of commands makes no difference to DOS or to the operation of your computer.

Order might make a difference, however, if you are installing specialized disk partitioning software or a disk compression utility, because you may have to set up your disk properly before DOS can find some of the files specified in the DEVICE or INSTALL commands. If DOS reports that it can't find a file for one of these commands while it is processing CONFIG.SYS, try moving the partitioning- or compression-software installation command to the beginning of the file.

The order also may make a difference if you are using 80286 or 386 memory management software (such as QRAM, QEMM, 386MAX, or 386EMM.SYS). If you want one of these programs to load another software file or DOS's buffers into high memory, you must make sure that the memory manager is installed first.

If you are using software that loads from your CONFIG.SYS file, check its documentation to see if installation order is important. Otherwise, organize the commands in your CONFIG.SYS file in any manner that makes sense to you.

Q How do I determine the optimum settings for FILES and BUFFERS?

A The FILES setting tells DOS the maximum number of files that you can have open at any one time. Large applications such as database managers sometimes can use 20 files or more at once. The documentation with each application should tell you the minimum or optimum value for FILES. Use the highest value requested by the applications you use.

If you use a memory-resident program that can read or write files when it pops up, you should increase the FILES setting by one or two. The same is true if you are apt to return to DOS temporarily ("shell" to DOS) from inside a major application and do anything more than look at a directory listing.

The BUFFERS setting is more difficult to pin down. It lets DOS use a rudimentary disk caching scheme and hold data in memory that an application is likely to need. Each buffer that you create with the BUFFERS command gives DOS more room for this cache. That does not mean, however, that more is necessarily better. Each buffer uses some memory, and if the setting for BUFFERS is higher than the setting for FILES, the extra memory simply is wasted. DOS never uses more than one buffer for each open file.

A good rule of thumb is to set BUFFERS somewhere between 50 percent and 100 percent of the FILES setting, but not higher than 20 or 30. You can experiment with different BUFFERS settings, if you wish, to see which values make your most disk-hungry applications run the fastest.

Q What is the purpose of the STACKS command, and when should I use it?

A Behind the scenes, your computer constantly stops to take care of hardware interruptions. For example, the internal clock ticks 18.2 times each second, and the computer must update the current time during each tick. The computer also is interrupted each time you press or release a key on the keyboard and during other hardware activities.

Each of these interruptions is handled by a small program in memory or ROM. And each of these programs needs a stack, a space in which to work. If you use DOS 3.1 or earlier, or if you use a STACKS=0,0 statement, the stack space is "borrowed" from the program that is running when the interruption occurs.

This setting usually is adequate, but occasionally it may cause your system to act erratically, lock up for no apparent reason, or report an "Internal stack failure" error message and quit. If that happens to you, consider changing the STACKS line in your CONFIG.SYS file.

If you don't specify a STACKS setting, DOS uses 9,128 as the default settings, on most machines. This means DOS creates nine stacks for hardware interrupts and gives each one 128 bytes of space. If you use an unusual hardware board and your computer seems unstable whenever that board is running, try allocating more stacks (increase the first number), larger stacks (increase the second number), or both.

Q I upgraded to DOS 5.0, and one of my applications now issues an "Incorrect DOS version" message whenever I try to run it. What is the problem, and can it be fixed?

A Most programs begin by checking the DOS version you are using. They do this because each version of DOS makes new services available to applications, and the applications want to be sure that the version of DOS running on your computer provides the necessary services. Most programs have internal logic that says something such as, "Get the DOS version. If the version number is less than 3.0, then display an error message and stop." A few programs use more restrictive logic: "Get the DOS version. If the version number is less than 3.0 or greater than 4.0, then stop." The latter types of programs are the ones that are apt to give you problems.

If you have an application that fits into this category, it may well run without difficulty with DOS 5.0 if only it can get past the version number check. DOS 5.0's two-part SETVER command will let you find out if this is indeed the case. First, make sure that your CONFIG.SYS file has the following line:

```
DEVICE=SETVER.EXE
```

This command tells DOS to load a table of application names and version numbers each time you boot up your computer.

Next, you must add the balky application to the table by typing:

```
SETVER C:\PATH\BALKY.EXE x.xx
```

at the DOS prompt, where *x.xx* is any valid DOS version number. Finally, reboot your computer to make the new line take effect.

It is important to understand that the SETVER command simply makes DOS 5.0 lie to selected applications when they ask for the DOS version number. It does not make DOS 5.0 mimic other versions of DOS. A very few utility programs work with undocumented DOS data structures and must change the way they behave for each version of DOS. Fooling such a program with the SETVER command may cause it to run amok because it will misinterpret the information that it finds in memory. The moral is to use the SETVER command with some caution and only when it is absolutely needed or when the distributor tells you to do so.

Q Sometimes my computer issues a "System halted, cannot load COMMAND.COM" error message. When this happens, I have to reboot the computer. What causes this error message, and what can I do about it?

A Technically speaking, COMMAND.COM is a shell, a program that interprets the commands you type, turning them into requests for services that DOS can understand. Many users think that COMMAND.COM is DOS because it is the program that is responsible for displaying the > prompt and waiting for user input.

Every time you exit an application program, a shell must be running, ready to accept and interpret your commands. Without the shell, your computer can't do anything because DOS does not know how to accept and interpret your commands, nor does it know how to run a batch file or even an application without the help of the shell.

COMMAND.COM itself has two parts. The "resident" portion is relatively small and stays in memory at all times. A "transient" portion is loaded into the top of available memory whenever the > prompt is available or a batch file is running. After an application ends, the resident portion of COMMAND.COM checks whether that application used the memory where the transient portion was stored. If so, it loads the transient portion from disk again before it displays the prompt.

If the loading procedure fails for some reason, the resident portion gives up, displays the error message you mention, and locks up the computer. After all, nothing can happen unless COMMAND.COM is running.

Many possible causes for this failure exist. The most frequent source of the problem is that the resident portion of COMMAND.COM cannot find itself on disk. This may happen if you boot from a floppy disk, move COMMAND.COM to a RAM disk for faster loading, and forget to tell the system that you have done so. To find out if this is the case, issue a SET command at the DOS prompt; then look at the line containing the COMSPEC statement. The line should not contain any spaces, and the spelling and pathname for COMMAND.COM should be correct (if you are using a replacement shell such as 4DOS, its name should apear in place of COMMAND.COM). If the pathname or filename isn't correct, put a line such as the following in your AUTOEXEC.BAT file:

```
SET COMSPEC=D:\COMMAND.COM
```

This tells COMMAND.COM where to find itself.

Another common reason for the load failure is that the last application has somehow destroyed the internal memory map that DOS maintains. If the COMSPEC setting is correct, and you can create this error consistently when you leave a specific application, you have found a legitimate bug in that program. Contact the publisher of the program and ask whether the company has an updated version that corrects the problem.

Q The PATH statement in my AUTOEXEC.BAT file is so long that it runs off the screen when I use a text editor to display AUTOEXEC.BAT on the screen. Can I split the PATH statement into two or more lines so I can see it all?

A You can split your PATH statement, but to understand the technique for doing this, it helps to know a bit more about PATH and DOS's environment space, a message pad area that DOS and other programs write to and read from.

When DOS reads a PATH command, it stores the information contained there in the environment, as a variable called Path. You can see this information by typing SET at the DOS prompt. You also can use the SET command to store information in a variable name. For example, you can use a command such as:

```
SET MYNAME=Jane Doe
```

to save your name in the environment space.

Beginning with DOS 3.0, COMMAND.COM can read an environment variable and place its value—the information after the equal sign—in any line in a batch file, including AUTOEXEC.BAT. To do this, you simply insert the variable name, such as Myname or Path, into the batch file command, placing it between two percent signs. For example, when you add the above SET command to your AUTOEXEC.BAT file and insert the following line near the end of AUTOEXEC.BAT:

```
ECHO Hello %MYNAME%
```

the computer will display this message at bootup:

```
Hello Jane Doe
```

The following two lines show how you might use this trick to break your PATH setting into two or more lines:

```
PATH C:\DOS;C:\WINDOWS
PATH %PATH%;C:\WORD
```

The first line inserts two subdirectory pathnames in the Path variable; the second adds another subdirectory to the list. Although you can use this technique to spread your PATH command over two or more lines, the total PATH setting still is restricted to 127 characters.

One danger exists, however. Many automatic installation programs try to add their application's subdirectory to your PATH. And most don't make allowances for AUTOEXEC.BAT files containing more than one PATH statement. Their strategy is to add their subdirectory name to every PATH or SET PATH line in AUTOEXEC.BAT. If this happens, you'll end up with PATH statements such as:

```
PATH C:\DOS;C:\WINDOWS;C:\XYZ
PATH %PATH%;C:\WORD;C:\XYZ
```

which means that the C:\XYZ directory will appear twice in your path. That won't hurt anything unless your path already is approaching its maximum length. To protect against such foolishness, you should always examine your CONFIG.SYS and AUTOEXEC.BAT files after you run an automatic installation program and correct any errors that may have occurred.

Q I use the DOSSHELL program that comes with MS-DOS 5.0, but I have noticed that if I reboot the computer or turn it off while still in the shell, I sometimes end up with strange-looking files on my hard drive. They have names such as 12E5DOSC.BAT, 11D4DOSC.BAT, and so on. Is this a problem? Can I safely delete these files?

A Many programs, including DOSSHELL and Windows, create temporary files that they intend to delete when they end. If you turn off your computer before shutting down these programs and returning to the DOS prompt, the temporary files stay on your hard disk. The moral: Always exit from all programs and return to the DOS prompt before you turn off your computer.

If you forget, or if your computer locks up and you must reboot, you can delete these temporary files. First make sure that you have exited from all of your programs, including Windows and DOSSHELL. If you delete a temporary file from a program that is still running—if for example, you shell to DOS or use Windows' DOS box and erase these temporary files—you can crash the program that is using the file.

One way to recognize temporary files is by their unusual names. Another is to examine the files' dates and times to ensure they were created on your machine (instead of shipped with the program) and that they were created before you last rebooted or turned on your computer. ■

A Six-Pack Of DOS Treasures

Enrich your computing repertoire—and improve your efficiency—with some of DOS's niftiest utility programs.

by John Wolfskill

Read your DOS manual carefully and you'll find, buried within its pages, descriptions of a dozen or more utility programs that can help save you frustration and increase the speed and efficiency of your computing by leaps and bounds. While it's not difficult to find these programs in your DOS directory, the trick is knowing when and how to use them to your best advantage.

In this article, I discuss six DOS utility "treasures" and offer practical advice on putting them to work. If you're new to MS-DOS computing, these programs offer a great way of getting your feet wet. If you're a seasoned DOS user, you may find in DOS 5.0 a new set of treasures to explore—not to mention the added challenge of harnessing expanded and extended memory. To take you one step further, I also cover five of the most frequent problems our readers encounter with DOS 5.0.

RAMDRIVE.SYS

High up on my list of indispensable utilities is RAMDRIVE.SYS (VDISK.SYS in PC-DOS), a device driver that lets you set aside a portion of memory (RAM) to act as a temporary disk drive. If your regular applications don't consume all of your system's memory, you can put some of it to good use by setting up a RAM disk (also called a virtual disk) in which to run programs that require frequent disk access. Because the information on a virtual disk resides in memory, your computer can access that information more quickly—up to 15 times faster than it can access data on ordinary disk drives.

But virtual disks do have a significant drawback. Each time you turn off your computer, you lose the data stored in memory. To avoid data loss, you must copy files to your virtual disk at the beginning of each operating session and save those files to one of your regular disk drives before you turn off your computer.

When should you use a virtual disk? The easiest way to spot a program that may benefit is to load the program and run it. If you frequently must wait while your computer stores and retrieves information on disk (you'll hear the drive swinging into action—all too often—or notice the drive light coming on a lot), the program probably will operate faster from a virtual disk.

However, unless your computer is equipped with several megabytes of expanded or extended memory, you may find that large applications, such as word processors, databases, or spreadsheets, will require more conventional memory than you can spare for a virtual disk. Moreover, sometimes the data files a program must read from your physical disk drives—rather than the program itself—are the cause of the bottleneck. All these factors mean that you may need to experiment to find out when a virtual disk is appropriate and how much memory you need to allocate to achieve maximum efficiency.

Setting up a RAM disk is straightforward. Like other device drivers, RAMDRIVE.SYS is invoked by adding a command line to your CONFIG.SYS file. For instance, to set up a 128K virtual disk in conventional memory, you might load your CONFIG.SYS file into your word processor and add the following line:

```
DEVICE=C:\DOS\RAMDRIVE.SYS 128
```

This command assumes that RAMDRIVE.SYS is located in a subdirectory named C:\DOS.

> ***If you frequently must wait while your computer stores and retrieves data on disk, that program probably will operate faster from a virtual disk.***

If your PC has expanded or extended memory, you should use it instead of conventional memory. The switches to do this are /A for expanded memory and /E for extended memory. For example, this command:

```
DEVICE=C:\DOS\RAMDRIVE.SYS 300 /A
```

sets up a 300K virtual disk in expanded memory.

By default, a virtual disk is formatted with 128-byte data sectors and space for 64 directory entries. If you need to modify the default settings, you can specify additional parameters. For example, to reserve space for 100 directory entries and a sector size of 512 bytes, type the following command:

```
DEVICE=C:\DOS\RAMDRIVE.SYS 512 100 300
```

Then, save the modified CONFIG.SYS file in text or ASCII format, exit your word processor, and press Ctrl-Alt-Del to restart your system. At bootup, DOS reads its CONFIG.SYS file and automatically installs the virtual disk, using the next available drive letter. If your system is equipped with one or two floppy drives (drives A and B) and a hard drive (drive C), DOS automatically assigns the logical letter D to your virtual disk. If no error messages appear while DOS installs RAMDRIVE.SYS, you're ready to use your virtual disk exactly as you would a physical disk.

Managing a virtual disk is more involved than installing one, however. After you identify the programs or data files you want to store on your virtual disk, you must copy them to the virtual disk at the start of each operating session. The best way to do this is with a DOS batch file. The short batch file below is the one I use to set up my virtual disk to hold the text files associated with my word processor:

```
@ECHO OFF
MD D:\PW
CD PW
COPY *.TXT D:\PW >NUL
PW
COPY D:\PW\*.TXT C:\PW >NUL
CD\
```

This batch file uses DOS's asterisk wildcard (*.TXT) to copy files between the C:\PW and D:\PW subdirectories. The wildcard makes it unnecessary to specify each file by name and thus provides a simple and reliable way to shuttle files between a physical and virtual disk—if the combined size of your files doesn't exceed the space you've allotted for your virtual disk.

My batch file copies to my virtual disk (drive D) all files ending with the extension .TXT; then it starts my word processor (PW). When I exit the program, the batch file automatically copies the same files back to my hard drive and returns to the DOS prompt. If you have several applications that might benefit from a virtual disk, you can set up individual subdirectories within the virtual disk to perform a similar function for each application. Adding >NUL to the lines that copy the files is simply a housekeeping measure. It keeps things neat by preventing the messages generated by the COPY commands from appearing on the screen.

MODE.COM

The MODE command provides an easy means of controlling settings for your printer, communications ports, keyboard, and video display. MODE.COM has been around since the introduction of DOS 1.0 but has undergone several changes that have increased its flexibility. Version 1.1 improved serial port communications, Version 4.0 added a parameter that lets you control the response rate of your keyboard's keys, and DOS 5.0 now provides a status parameter that displays a list of all devices affected by MODE and the current status of those devices.

MODE is useful in a variety of situations. For example, it provides a way to configure your serial communications port directly from the DOS command line. The proper syntax for doing this is:

MODE [COM*n*],[*transmission rate*],
[*parity*],[*data bits*],[*stop bits*],[P]

where *COMn* is a valid serial device from COM1 through COM4. Valid settings for transmission rate are 110, 150, 300, 600, 1200, 2400, 9600, or 19,200 bits per second (bps). The error-checking parity bit may be specified as *N* (none), *O* (odd), or *E* (even). The number of data bits per word may be set to either 7 or 8, and the number of stop bits may be 1 or 2.

The P parameter tells DOS to keep trying to send output to the port, even though the port may be busy. If you add this parameter to the end of the command, DOS loads a portion of MODE.COM as a memory-resident program. DOS 5.0 users can specify the updated retry parameter instead of the P parameter, where the retry value is set as *E* (return an error message), *B* (return a busy status), *P* (retry infinitely), *R* (return a ready status), or *N* (no retries).

As the brackets indicate, most of elements in the command are optional. When you omit a parameter, DOS uses default values for those settings. If, for instance, you type the following command:

```
MODE COM1 1200
```

DOS initializes the transmission rate to 1200 bps, then uses its default settings (E,7,1) to fill in the blanks.

The parameters that follow the bps setting are position-dependent. If you skip any setting in the sequence, you must use a comma as a placeholder to avoid unexpected results. For example, the three commas in this command:

```
MODE COM1 1200,,,2
```

indicate that the value 2 refers to the number of stop bits.

MODE also lets you redirect the output of your parallel printer port to a serial communications port. This feature is handy when you're using a program that assumes your printer is connected to LPT1 and it actually is connected to COM1. The command to redirect parallel port output is:

```
MODE [LPTn] = [COMn]
```

where *n* is any port value from 1 to 4.

With this variation on MODE, you can use a DOS batch file to initialize your serial port, redirect printer output to your serial port, load and run the program that assumes your printer is connected to LPT1, then cancel the redirection. Here is a DOS batch file that will do the job:

```
@ECHO OFF
MODE COM1 1200,N,8,1
MODE LPT1=COM1
REM Substitute the name of your program
REM for PW in the following line.
PW
MODE LPT1
```

Before you try this out, however, check your printer's manual to establish its default serial communications settings.

DOS 5.0 users also can take advantage of MODE's updated /STATUS switch to display a list of the current settings for all ports by typing:

```
MODE /STATUS
```

Most printers will respond to the MODE command's request to set the printed line width and the number of printed lines per inch. The former is useful when you must control two printers through a remote data switch box and their carriage widths differ. The appropriate syntax is:

```
MODE [port][characters per line]
     [lines per inch][P]
```

where a valid port number is LPT1 through LPT4, the number of characters per line may be 80 or 132, the number of lines per inch may be 6 or 8. The default values are LPT1, 80, and 6. For example, this command:

```
MODE LPT1 132 8 P
```

sets the printer to 132 characters per inch, 8 characters per line, and continually tries to send output to a remote data switch box until a port is available to receive data. (You can press Ctrl-C to abort attempts to see if the port is free and thus regain control of your computer.)

If you use DOS 4.0 or later, you can use the MODE command to set the "typematic" rate of your keyboard, the speed at which DOS repeats a keyboard character when you hold down a key. You also can set the time delay DOS imposes before it begins to repeat a character. The MODE command line for this feature is:

```
MODE [rate][delay]
```

where *rate* is a value from 1 to 32 (the default value is 20), and *delay* is the number of quarter-second increments from 1 to 4 (the default value is 2). For example, the command:

```
MODE 32 1
```

sets the typematic rate to maximum speed, and the time delay to the minimum increment.

The MODE feature is handy when you're using an application that assumes your printer is at LPT1 and it actually is connected to your COM1 port.

Making the delay setting higher than the default value is useful for heavy-handed typists who tend to rest their fingers on the keys while pondering a problem. Be careful, though, not to set the rate too high. A delay value of 4 (one second) usually causes sluggish operation when you use the space bar to move the cursor.

The most common—and the most misunderstood—use of the MODE command is to set the video display width and display mode. The MODE command doesn't allow you to change from one text or graphics display mode to another. Instead, it determines whether your video adapter will display its output in color or black and white and automatically shifts your display adapter to DOS's default text mode. The MODE command line for this purpose is:

```
MODE [display][shift][T]
```

where *display* can be 40, 80, BW40, BW80, CO40, CO80, or MONO. The numbers 40 and 80 indicate a displayed output of 40 or 80 characters per line. The letters *BW* indicate a black-and-white display, and the letters *CO* indicate a color display.

The MONO parameter lets you shift between color and monochrome monitors installed on the same system. For example, if your system is equipped with both VGA and monochrome display adapters, you can shift to a monochrome display by typing:

```
MODE MONO
```

then return to the VGA display by typing:

```
MODE CO80
```

> ***After you install DOS 5.0's SMARTDRV.SYS disk-caching program, your computer will spend much less time reading information in from your system's hard drive.***

The second parameter lets you shift the horizontal screen position left or right in pixel increments by specifying the letters *L* or *R*. This parameter is intended for use with an IBM CGA video adapter operating in composite-video (TV) mode. The T parameter displays a test pattern that aids in centering the display.

Although the shift parameter may work with some fully compatible EGA or VGA display adapters, it's easier and faster to center the display by moving your monitor's horizontal screen position knob.

ANSI.SYS

Introduced in DOS 2.0, ANSI.SYS has remained virtually unchanged since then. This command provides the only way for nonprogrammers to alter screen and text colors, control the movement and location of the cursor, and reassign the functions of key sequences.

The syntax for loading ANSI.SYS in your CONFIG.SYS file is similar to that for other device drivers:

```
DEVICE=C:\DOS\ANSI.SYS
```

After pressing Ctrl-Alt-Del to reboot your computer, you are ready to experiment with ANSI.SYS commands.

Once installed, the ANSI.SYS keyboard driver code examines every keystroke typed at the keyboard. When it encounters the appropriate keystroke combination—the ASCII escape character followed by a left square bracket—it leaps into action. On screen, the special wake-up sequence looks like a left arrow (ASCII 27) followed by an open bracket (ASCII 91):

```
<–[
```

Each time the ANSI.SYS driver spots this special combination, it interprets the keystrokes that follow as an ANSI escape sequence.

Unfortunately, you cannot type the escape character directly from the DOS command line, so it's not possible to simply type the ANSI driver commands. Although you may follow several methods for creating the escape character using your word processor or text editor, the DOS PROMPT command provides the most convenient way to send ANSI sequences to the keyboard. With the PROMPT command, you can specify the escape character using the characters $e. For example:

```
PROMPT $e[45;33m$P$G
```

changes the foreground color of a text screen to yellow and the background text color to magenta, then creates a DOS prompt that shows the current drive, followed by a greater than (>) sign.

These are but a small sampling of the types of things that ANSI.SYS can do. Your DOS manual should contain a complete list of keystroke sequences to which ANSI.SYS will respond. As you experiment, keep in mind that the total space available to store commands in the ANSI.SYS data storage buffer is less than 1024 bytes.

GRAPHICS.COM

Every now and then, it's handy to be able to quickly print out a copy of the information you see on screen. If you're working with a text screen, pressing the PrScr key usually does the trick. PrScr won't work for graphics screens, but DOS's GRAPHICS.COM will. Once loaded, this memory-resident utility hovers in the background, waiting for instructions telling it to print the contents of your graphics screen display. GRAPHICS.COM works in all CGA, EGA and VGA graphics modes.

To load the program into memory, simply type:

```
GRAPHICS
```

at the DOS prompt. Thereafter, you need only press Shift-PrScr to send a graphics-mode screen to your printer. Because the memory-resident portion of GRAPHICS.COM requires only 8.5K of conventional memory, you may want to consider loading it along with AUTOEXEC.BAT each time your computer starts. The program remains active in memory until you reset your computer.

Since it was first released with DOS 2.0, GRAPHICS.COM has undergone a quiet face-lift with every major DOS upgrade. DOS 3.0 added support for non-*IBM* printers, while DOS 4.0 added support for color printers and improved support for the IBM VGA. Version 5.0 includes support for most of the Hewlett-Packard LaserJet printer family, but it doesn't accommodate PostScript printers.

The program uses a data file called GRAPHICS.PRO to define the characteristics of the printers it supports. Although you can modify this text file to support other printers, the file doesn't appear to be intended for that purpose.

SMARTDRV.SYS

If you use DOS 5.0, you'll find the SMARTDRV.SYS disk-caching program a welcome addition to your operating environment. After you install the program, your computer will spend less time reading data from your hard disk, and you'll spend less time waiting for program output. In fact, a disk-caching utility is one of the best ways to increase the overall speed and efficiency of your computer.

SMARTDRV.SYS works its magic by setting aside a portion of your computer's expanded or extended memory as a place to temporarily store information that may be needed in the future. Because it reduces the time required to retrieve information from disk, your programs gain access to it more quickly.

Although we think of data as numbers, records, or strings of characters, DOS stores it on your hard disk in chunks called disk sectors. When your program writes a file to disk, DOS writes the information to empty disk sectors, which it allocates in sequential fashion. DOS first allocates a group of empty disk sectors, then fills them with data until the entire file is stored on disk. After recording the file, the sector numbers associated with the file are stored in the file-allocation table (FAT), a data table DOS uses to keep track of the storage areas assigned to each file.

A disk-caching utility works under the assumption that information retrieved from your hard disk may be required by your program again. It also assumes that the next data you'll need is stored on disk in the same general vicinity as the data most recently read.

Each time your program reads information from your hard disk, SMARTDRV.SYS stores a copy of the transaction in a special memory area called a cache. When your program requires new information, SMARTDRV.SYS first checks to see if it already is in the cache. If so, it immediately hands off the data to DOS, which eliminates the comparatively long and involved process required to locate the information on your computer's hard drive.

An example of a rudimentary disk cache is the DOS BUFFERS command, which you can place in your computer's CONFIG.SYS file. When you specify BUFFERS=30, DOS allocates memory space for 30 disk sectors as a pseudocache. Unfortunately, the BUFFERS command is not intelligent: It can't tell the difference between a sector that contains often-used directory information and one that contains data you'll rarely use again.

SMARTDRV.SYS provides the basic services of an intelligent disk-caching program, although it is not as sophisticated as some third-party utilities. You can install it in expanded or extended memory by adding a DEVICE command to your CONFIG.SYS file:

```
DEVICE=C:\DOS\SMARTDRV.SYS
[cache size] [/A]
```

where *cache size* is the number of kilobytes of memory you want to allocate. The /A switch tells DOS to set up your cache in expanded memory. Without the /A parameter, SMARTDRV.SYS creates your cache in extended memory. For example, to install SMARTDRV.SYS in extended memory, you would add the following lines to your AUTOEXEC.BAT file:

```
DEVICE=C:\DOS\HIMEM.SYS
DEVICE=C:\DOS\SMARTDRV.SYS 1024
```

If you plan to use expanded memory for your cache, make certain you place the DEVICE command that installs SMARTDRV.SYS after the line that installs your expanded memory manager:

```
DEVICE=C:\DOS\HIMEM.SYS
REM - DEVICE command to install
REM - your EMM goes here.
DEVICE=C:\DOS\SMARTDRV.SYS 1024 /A
```

> ***DOSKEY, probably the best new utility offered in DOS 5.0, provides a way for you to view, edit, and carry out DOS commands that you have issued previously.***

When you've modified CONFIG.SYS, reboot your computer to activate SMARTDRV.SYS.

Unlike the BUFFERS command, SMARTDRV.SYS reads and stores the contents of an entire disk "track" each time your program reads information from a disk sector contained on (or near) the track. (A disk track comprises a number of adjacent disk sectors that form a circle on your hard drive's data platter.) The reason SMARTDRV.SYS reads an entire disk track is that, as mentioned earlier, it expects to find information you'll need in the future on one of the sectors adjacent to the one just retrieved from your hard disk.

What is the best size for a disk cache? That depends on the programs you use and the amount of expanded or extended memory available in your computer. As a rule, a disk cache of less than 256K, the minimum allotment, won't be effective for programs that store and retrieve large data files, such as desktop publishing files, or for graphics-intensive environments such as Microsoft Windows 3.0.

The best way to determine the optimum size for your disk cache is to install a 256K cache, then load and run the programs you normally use. Observe your hard disk activity light as each program runs. If the light blinks continually during normal periods of disk access (after the data has been initially read from disk), your cache may be too small. Increase the size of your cache and run the program again. Of course, you won't be able to eliminate all hard disk activity. The best you can hope for is to minimize disk activity in all of your programs.

DOSKEY.COM

DOSKEY.COM is perhaps the best new utility offered in DOS 5.0. It provides a way to view, edit, and carry out DOS commands you've used previously. You also can create, run, and save command macros, a command sequence consisting of one or more stored DOS commands. In effect, a DOSKEY macro works in the same way as a DOS batch file. However, DOSKEY macros are stored in memory, so they execute many times faster than batch files. Also, you don't need to load a text editor to create them. A disadvantage is that DOSKEY macros disappear when you reset your PC, unless you save them as a file.

DOSKEY.COM requires 3K of memory in addition to its command storage buffer. You can load the memory-resident utility by typing its filename at the DOS prompt, or you can include it as an item in your AUTOEXEC.BAT file. The following command line, for instance, tells DOSKEY to reserve a storage buffer of 2048 bytes:

```
C:\DOS\DOSKEY /bufsize=2048 /H /I
```

(the default buffer size is 256 bytes). Including the /H parameter lets you use the arrow keys to display a history of the most recently typed commands. Adding /I sets the built-in command editor to use the insert mode, making it possible to set right to work adding new text to old commands. To initiate overstrike mode, you would use /O instead. Unlike the awkward "command template" found in older DOS versions, DOSKEY provides a simple method for editing a command line.

The syntax for creating a macro is:

```
DOSKEY [macro name]=[command1]
  $T [command2]...
```

where *macro name* is a name you assign and the commands are any valid DOS commands, batch file names, or executable program names; and $T is a separator that marks the end of individual commands. The total length of a DOSKEY command line may not exceed 127 characters.

As with DOS batch files, you can use replaceable parameters to specify drive letters and so on by using variables, but instead of %1 through %9, DOSKEY uses $1 through $9. For example, typing:

```
DOSKEY D=DIR $1
```

creates a macro called D. If you then type this command:

```
D A:
```

DOS provides a directory of a floppy disk in drive A. Although this shortcut may not seem spectacular, it saves a couple of keystrokes. A better example of DOSKEY's potential is a macro command I call MCOPY:

```
DOSKEY MCOPY=CD\ $T MD $1 $T COPY
  \$2\*.* \$1 $T CD$ 1 $T DIR
```

This macro logs onto your hard drive's root directory, creates a subdirectory ($1), copies all of the files contained in another subdirectory ($2) to the new subdirectory, logs onto the new subdirectory, then displays a directory listing. All of this occurs automatically when you invoke MCOPY and name the two subdirectories. For example, typing:

```
MCOPY NEWDOS DOS
```

names C:\NEWDOS as the directory to be created and C:\DOS as the directory containing the files to be copied, assuming you are in the root directory of drive C.

DOS 5.0's ability to save DOSKEY macros as a DOS batch file adds even more flexibility. For example, you can create one or more macro commands, then save the macro or macros as a DOS batch file by typing the following command:

```
DOSKEY /MACROS > FILENAME.BAT
```

If the above macro called MCOPY is in the macro buffer when you issue this command, DOSKEY creates a one-line DOS batch file that reads:

```
MCOPY=CD\ $T MD $1 $T COPY \$2\*.* \
  $1 $T CD$ 1 $T DIR
```

To save the macro definition for later use, you simply load the batch file into your text editor and add to the beginning of the macro line a call to install DOSKEY. That line will then read:

```
DOSKEY MCOPY=CD\ $T MD $1 $T COPY
  \$2\*.* \$1 $T CD$ 1 $T DIR
```

Finally, you must save the modified batch file. If you then include the modified file as an item in your AUTOEXEC.BAT file, DOS will install the macro contained in the batch file each time you start your computer.

John Wolfskill is a free-lance writer and president of Powertrain Development, a New Hampshire-based software development company.

Questions & Answers

IMPROVING PRODUCTIVITY

Q Which DOS files can I delete from my hard disk or floppy disk to conserve disk space?

A The number and types of DOS files you can safely delete from your hard disk depend on your DOS version, the type of computer you use, and the way you use it. Only three DOS files are required to provide core file management services (such as copying, deleting, renaming, and listing of files) and to run application programs. The files are IO.SYS, MSDOS.SYS, and COMMAND.COM (the first two files are called IBMBIO.COM and IBMDOS.COM in PC-DOS).

You should never delete these files from your disk. However, all remaining files are fair game. The best way to decide which ones you can afford to remove from your disk is to consult your DOS manual. There you'll find each command listed, along with a description of the purpose of the command, proper syntax, comments about the command, and examples of how to use it.

DOS commands are classified as either internal or external commands. Because all internal commands are included in COMMAND.COM, this leaves only external commands from which to choose. External commands are individual files that provide DOS utility programs, device drivers, or programming tools.

After you read the description of each external command, decide if you might ever need it in your daily work. Make a list of those external files that appear superfluous, then delete them from your DOS subdirectory. If you have DOS 4.x and earlier, restoring a purged external file is relatively easy. You simply copy the file to your hard disk's DOS subdirectory from the original system disk.

DOS 5.0 users will find the restoration process a bit more involved. Starting with DOS 5.0, system files are stored on the original system disks in compressed format. The best way to ensure the files you've deleted are accessible when you need them is to create a second copy of DOS 5.0 on floppy disks (use the Setup program), then put the disks aside for safekeeping.

If your work doesn't involve international correspondence, you won't need the DOS files NLSFUNC.EXE, GRAFTABL.COM, KEYB.COM, COUNTRY.SYS, DISPLAY.SYS, KEYBOARD.SYS, PRINTER.SYS, or any of the files that end with the .CPI file extension. If you have DOS 5.0 and your computer is equipped with an 8086, 8088, or 80286 CPU, you can delete EMM386.EXE, the upper-memory manager for 80386 class computers. If you don't plan to add virtual disks to your system, or install a disk cache to speed up your hard disk drive, you also can purge RAMDRIVE.SYS (or VDISK.SYS) and SMARTDRV.SYS

If you're not a computer programmer, you can strike EXE2BIN.COM, GWBASIC.EXE, BASICA.COM, QBASIC.EXE, QBASIC.HLP, and all files with the file extension .BAS. However, if you use DOS 5.0, and you also plan to use its text editor, EDIT.COM, do not delete QBASIC.EXE. These programs share the same code.

DOS 4.x and 5.0 users who don't need the services of the DOSSHELL program can delete DOSSHELL.EXE, DOSSHELL.COM, DOSSHELL.GRB, DOSSHELL.VID, DOSSHELL.HLP, and DOSSHELL.INI. All DOS users can safely delete the files called PACKING.LST and README.TXT.

Q I'm having trouble with a batch file I have written. The batch file consists of the following line of code:

```
DIR %1 | SORT | MORE
```

When I use it to look at a directory, I sometimes see a couple of odd looking filenames listed, such as:

```
AKDEJLB1    0 10:13:56p
AKDEJKB2    0 10:13:56p
```

The same files are not listed when I use a regular DIR command. Furthermore, when I try to use my batch file on a write-protected disk, I get the message "Error writing to device," and the batch file doesn't work. I can't figure out what is going on.

A The mystery files are temporary files DOS creates during the pipe (|) operation. Ordinarily, DOS deletes these files when the operation is complete, so you normally don't see them. Before DOS creates temporary files (in this case for internal use), it first looks in the DOS environment space for a variable you've named Temp, to which you've assigned a DOS pathname. If it finds that a valid pathname is assigned to Temp, DOS creates the temporary files on the specified drive. Otherwise, it creates the files on the current subdirectory (DOS 5.0 only) or the root directory (other DOS versions) of the current drive.

Your batch file failed to operate

properly because DOS could create the temporary files but could not write information to them. The zero length of the files indicates your floppy disk may have been write-protected. To avert the problem, add the following line to your AUTOEXEC.BAT file:

```
SET TEMP=C:\TEMP
```

to force DOS to write all temporary files to your hard drive instead of the write-protected floppy disk.

Q I'm trying to speed up the time it takes to access my hard disk drive. The DOS manual says the BUFFERS and FASTOPEN commands will do this for me, but I know disk-caching programs such as SMARTDRV.SYS and third-party utilities such as PC-Kwik also have this capability. Do BUFFERS, FASTOPEN, and disk caches all do the same thing? How do I determine the best setup for my machine?

A All of the solutions you mention will speed up the overall operation of your computer. The FASTOPEN command is a specialized type of disk cache that stores only directory information. FASTOPEN is primarily intended to help DOS speed up searches for directories. Therefore, it is most effective when your disk has many deeply nested subdirectories that your application programs access frequently. If you don't have DOS memory to spare, you may find FASTOPEN carries a fairly large RAM overhead for the limited services it provides.

The DOS BUFFERS command is a sector-based disk cache that allocates a portion of your computer's memory to hold the most recent information read from a storage device. BUFFERS is most useful for those who need a rudimentary disk cache but also need to conserve as much conventional memory as possible to run application programs. Each disk sector buffer uses 512 bytes of memory. Therefore, a typical setting of BUFFERS=20 uses about 10K of your computer's memory to provide a moderate performance boost.

If your primary interest is revving up your hard drive access for disk-intensive environments such as Microsoft Windows, I don't recommend BUFFERS as your only weapon. First, it's not as efficient as a "track buffering" cache such as SMARTDRV.SYS. The BUFFERS command works with smaller disk storage units called disk sectors, instead of entire disk tracks. The idea of a disk cache is to reuse the maximum amount of information your computer already has read from disk, so a track buffering cache will usually be more efficient.

Second, the BUFFERS command is difficult to fine-tune for a wide range of applications. If you allocate two few disk buffers, it will be ineffective. Conversely, if you allocate too many buffers, DOS will waste time searching for information that may not be in the cache. Finally, the BUFFERS command is not intelligent; that is, it doesn't know the difference between a sector that contains often-used directory information and one that contains data your application rarely will need again.

If you have the cash and memory to spare, the best overall solution is a third-party track buffering utility that caches both disk reads and writes. (The current version of SMARTDRV.SYS caches only disk reads.)

Q I need a way to compare two files to find out if they are exactly alike. Can this be done using DOS?

A Yes. You can use the external DOS file COMP.COM to compare two data or program files. COMP does not give you a detailed report of each place where the files differ, however; it reports only whether or not the files are identical and indicates the byte position within the files where the first variation occurs. Also, you can't use it to compare files of different sizes.

The easiest way to use COMP is to specify the two files you want to compare on the command line. If the files you want to compare are located in different directories, you must specify the entire DOS pathname. For example, this line compares a file or drive C with one on drive A:

```
COMP C:\PW\FILE1.TXT A:\FILE2.TXT
```

If the two files are identical, COMP will display a "File compare OK" message. Otherwise, it reports the first position where the files differ:

```
Compare error at offset D
file1 = 41
file2 = 45
```

If more than 10 mismatches are found, you'll receive the following message:

```
10 mismatches - ending compare
```

COMP.COM is not an elegant DOS utility. If you're not an experienced computer programmer, you may find it oddly convoluted. The word *error* in the message above doesn't indicate that it has encountered a program malfunction. It's simply a poor choice of words. The message indicates that different information was found at byte position 0D hexadecimal (13 decimal) as the files were compared. The numbers immediately below it indicate the hex value of the character found at offset 0D hex in both files; in this case, the ASCII characters A (41 hex) and E (45 hex).

Another message that may pop up is, "EOF mark not found," which should also not be a cause for alarm. It's a way of informing you that the program that created the file didn't end the file with a traditional ASCII 26 (1A hex) end-of-file character. In most cases, you can safely ignore the message. COMP.COM establishes a file's length by examining its directory entry, not by counting bytes until it reaches the EOF marker.

Q I recently ran CHKDSK on my computer and received the following message:

```
Errors found, F parameter not specified.
Corrections will not be written to disk
5 lost clusters found in 5 chains.
Convert lost chains to files (y/n)?
```

What should I do now? Does this mean I've lost some data from my hard drive?

A CHKDSK is a DOS utility that examines a hard disk or floppy disk to check the integrity of the file storage structure. It works by examining the file-allocation table (FAT) and every directory and subdirectory entry to make sure that each file is stored in a unique place on the disk and that each allocated storage area is associated with an entry in a directory.

You can run CHKDSK in two ways. If you type:

```
CHKDSK
```

the program will report any problems that it finds but will not make any corrections to your disk. If you type the following:

```
CHKDSK /F
```

the program will give you the choice of making corrections if it finds any errors. Unfortunately, it asks the same questions no matter which way you invoke the command. It simply ignores your response to its prompt for fixing an error if you type the command without /F.

CHKDSK can find and fix two kinds of errors. Lost clusters, such as those mentioned in your question, are sections of storage that aren't related to any filename. These occur most often if you shut down or reboot your computer without returning to the DOS prompt from an application—or without saving your files. Internally, DOS has allocated space for file data and has stored that data but hasn't yet updated the directory entries for the files to connect the data with the filename. This is not a DOS bug but rather the result of sloppy computing practices. Always return to the DOS prompt before you turn off your computer.

The other kind of error is called cross-linking and happens when two files seem to be claiming the same storage space. This could happen because of an internal DOS error but is more often caused by a buggy memory-resident utility or application that gets mixed up about what files it should be using.

In both cases, you really have only one choice. If you run CHKDSK and DOS displays an error message, type:

```
CHKDSK /F
```

and answer *Y* to the prompt. CHKDSK will collect the lost or cross-linked data into files, storing those files in the root directory of the disk. The filenames will be FILEnnnn.CHK, where *nnnn* is a number such as 0001.

These .CHK files may contain important data. You can look at them with the TYPE command or with a text editor or sector editor. If you want to keep the file, rename it to something you will remember. You could also extract the relevant parts with an editor and save them to a new file.

Delete the .CHK files to release their disk space back to the system when you are done looking at them and reclaiming any lost data that they may contain. Otherwise, they will simply use up valuable disk space. ■

Tip 4

Familiarity Breeds Contentment

Are you particular about the way you want your files organized when you request a directory listing? If so, and if you've upgraded to MS-DOS 5.0, you're in luck. DOS 5.0's new DIRCMD variable lets you specify a format for directory listings. Including the following line in your AUTOEXEC.BAT file, for example, tells DOS to group your files in alphabetical order by file extension and filename whenever you type in the DIR command at the DOS prompt:

```
SET DIRCMD=/OEN
```

You can include any valid DIR parameter in the DIRCMD variable. For example, to break long directory listings into groups of 22 files, you can add the /P parameter to the previous DIRCMD command. This causes DIR to pause each time the screen is filled and wait for you to press a key.

MEMORY MANAGEMENT

Sorting Out Memory

Making sense of expanded memory and extended memory isn't as hard as you think. But understanding the differences is extremely important.

by Brett Glass

EMS, XMS, UMBs, VCPI: The sea of acronyms surrounding memory management on IBM PC compatibles can be daunting to nontechnical users. Fortunately, managing your memory isn't all that difficult—if you know the basic vocabulary and use the right tools. It also helps to understand some of the history surrounding the development of the various types of memory.

In the Beginning...

In the beginning (as the story goes), there was the IBM PC—which came standard with an 8088 central processor, 16K or 64K of memory, a 160K floppy disk drive (if you didn't opt to use cassettes instead), and BASIC. (DOS cost you extra, and you had a choice between PC-DOS and Digital Research's CPM/86.) The PC was comparable in speed and power to the hobbyist computers of the day, and no one—least of all IBM—expected its architecture to dominate the computing landscape as it does today. In keeping with its humble origins (and the exorbitant price of RAM in those days—$200 for 16K), the IBM PC was designed to accept a maximum of 512K of RAM. (Later models bumped this limit up to 640K—the maximum possible without changing the architecture.) But as users of Lotus 1-2-3 and other memory-hungry products found themselves running out of RAM for large spreadsheets, graphs, and databases, they began to clamor for a way to get more.

Expanded memory was the first popular solution to this problem. The Expanded Memory Specification (EMS) was designed to allow any IBM-compatible computer, including those with the 8088 microprocessor, to access more than 1MB of memory for large applications. It works by switching blocks of RAM in and out of the processor's address space (see "Expanded Memory").

Versions of EMS before 4.0 could swap memory in and out of only a 64K range of addresses called a page frame, and could make four 16K pages of memory visible to the program at any one time. This required a lot of shuffling and made EMS difficult to use when program code, rather than data, needed to be swapped in and out. But EMS 4.0, which added innovations developed by AST Research, goes further: It allows any portion of RAM to be switched with any other. Because Lotus, Intel, and Microsoft were the first companies to promote this technique, EMS 4.0 and later frequently are referred to as LIM EMS.

Expanded memory has remained an important standard in the PC community. However, programmers find it awkward to view memory as a bunch of 16K pages. They're also eager to use the features of newer processors—especially their ability to handle huge amounts of memory gracefully. For this reason, EMS is gradually fading into the background.

Extended memory has arisen to take the place of expanded memory and is becoming the method of choice for breaking the 640K barrier.

The IBM PC/AT was the first PC-compatible computer to offer 1MB of memory without EMS. The "brain" of the AT, the Intel 80286 processor (usually called just the 286), had the ability to handle 16MB of memory—16 times as much as the original PC. However, for compatibility reasons, IBM did not provide any software to take advantage of this memory—except for its VDISK.SYS RAM drive utility (see "A Six-Pack of DOS Treasures" on page 29). And software developers, realizing that many more IBM PCs and compatibles than ATs had been sold, continued to support EMS, which could be made to work on the AT as well. Finally, extended memory was available only if the processor was switched into an advanced mode called protected mode, which few programmers understood and was not available on the original PC. Thus, the extended memory capabilities of the AT remained latent for years, used only by a few Ada compilers, RAM disks, and hard disk caching programs.

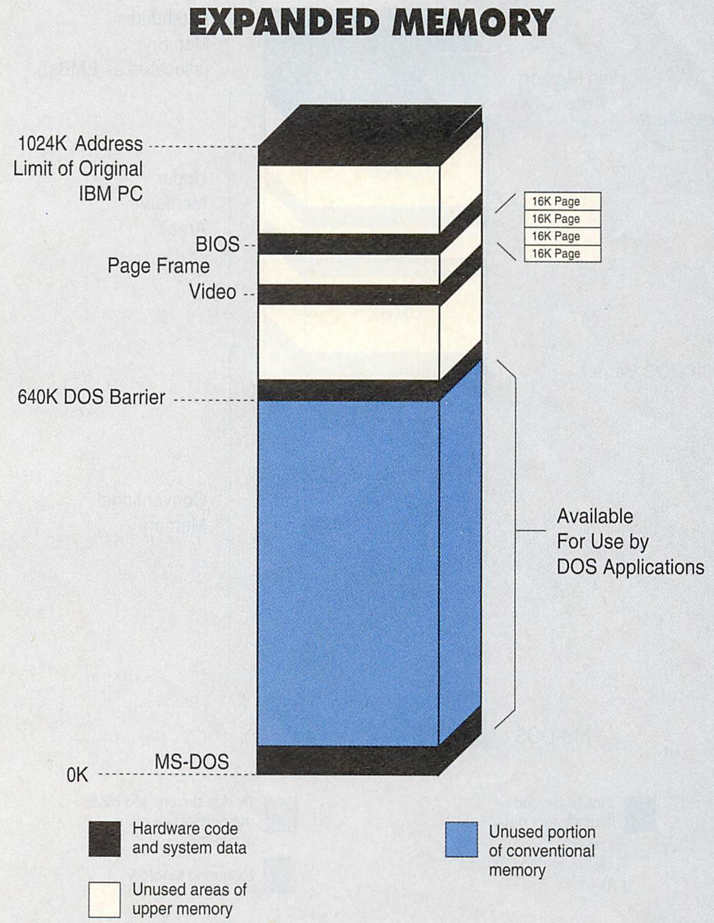
Not until the late 1980s—when 286-based systems surpassed older systems in market share—did software developers see the virtues of extended memory and begin using it instead of, or as an alternative to, expanded memory. Multitasking environments such as IBM's OS/2, Quarterdeck Office Systems' Desqview, and Microsoft Windows/286 and Windows/386 worked best with extended memory, as did memory managers for the new 80386. DOS extenders—tools that let programmers write applications for the protected modes of the 286 and 386—proliferated. Windows 3.0 took things a step further, requiring extended memory—rather than expanded memory—to run in standard mode and 386 enhanced mode.

At first, the industry lacked an adequate standard that would allow programs such as disk caches, RAM disks, and multitaskers to share extended memory gracefully.

The Extended Memory Specification (XMS), introduced by Microsoft, makes it easier for programs to share memory. To use XMS, you load an extended memory manager such as DOS 5.0's HIMEM.SYS, which reserves all of the extended memory in the machine and parcels it out to other programs (see "Extended Memory"). XMS lets programs reserve blocks of extended memory (extended memory blocks, or EMBs) and run programs from them. It also manages a 64K area of extended memory called the high memory area (HMA), which, due to an oversight by the designers of the Intel 286 chip, is available to programs without requiring them to switch to protected mode. Finally, it allocates upper memory blocks (UMBs), blocks of memory that fall between the 640K limit of conventional memory and 1MB. If your computer can map memory into this otherwise unused area (using EMS 4.0), you can load memory-resident programs (also called terminate-and-stay-resident programs or TSRs) and device drivers into UMBs.

Practical Memory Management

If you're interested in applying memory-management techniques to your PC, you probably have one of three goals in mind. You may want to load

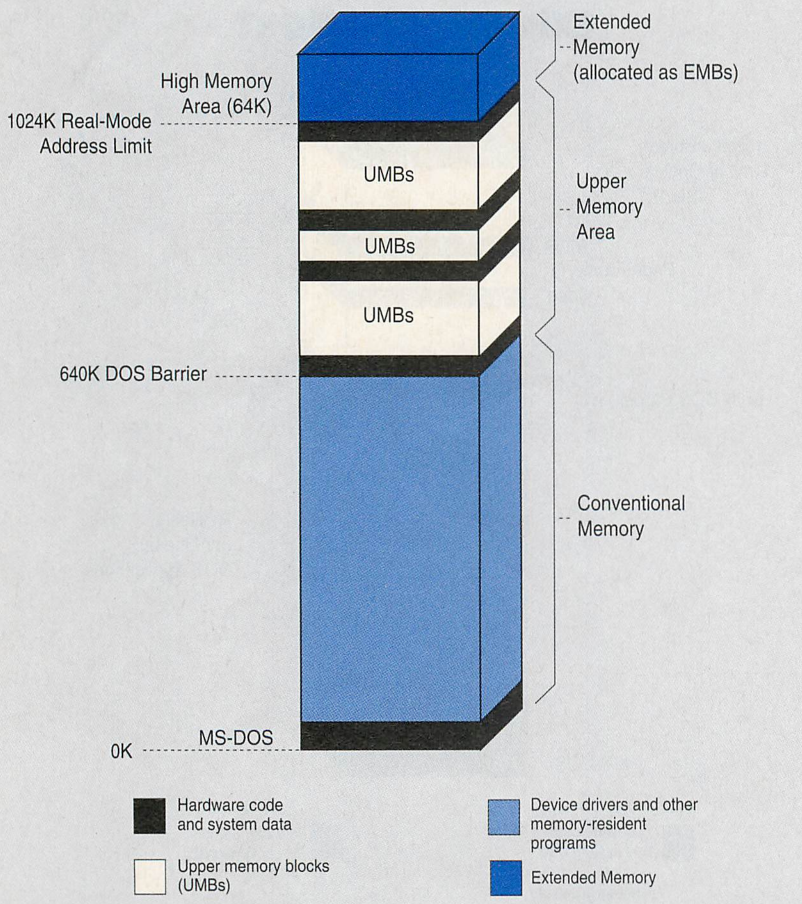

EXPANDED MEMORY
1024K Address Limit of Original IBM PC
BIOS
Page Frame
Video
640K DOS Barrier
0K
MS-DOS
16K Page
16K Page
16K Page
16K Page
Available For Use by DOS Applications
Hardware code and system data
Unused areas of upper memory
Unused portion of conventional memory

Expanded memory expands memory by placing only four pages of additional memory within view of the processor at any one time—just as very neat workers might have many pieces of paper but allow no more than four of them on their desks at a time.

TSRs, drivers, and the operating system into areas other than the lower 640K—to make way for programs that need lots of RAM. You might want to use expanded memory, extended memory, or some combination of the two to support programs that require them—including Windows (which needs XMS-controlled extended memory to run in standard or 386 enhanced mode). Or you may want to run programs that use DOS extenders—programs such as advanced versions of AutoCAD or the latest versions of Lotus 1-2-3.

Loading programs and the operating system high is your first option. To move MS-DOS 5.0 into the HMA three conditions must be met. First, you must have a 286 or 386 microprocessor and some extended memory. (All up-to-date PCs, and many as old as the original IBM PC/AT, have both.) Second, you need to have an XMS driver (such as HIMEM.SYS), or any third-party memory manager (such as Quarterdeck's QEMM386). Finally, you need to include a DOS=HIGH command in your CONFIG.SYS file. When these three conditions are met, MS-DOS 5.0 automatically uses the HMA, freeing up space in conventional memory. The setup program for Digital Research's DR DOS also can load programs into high memory.

EXTENDED MEMORY


Extended Memory (allocated as EMBs)
High Memory Area (64K)
1024K Real-Mode Address Limit
UMBs
UMBs
UMBs
Upper Memory Area
640K DOS Barrier
Conventional Memory
MS-DOS
0K
Hardware code and system data
Device drivers and other memory-resident programs
Upper memory blocks (UMBs)
Extended Memory

Extended memory is the area above the 1MB mark on a 286- or 386-based PC. It is simply a linear extension of the RAM below 1MB. A program such as HIMEM.SYS, called an XMS driver, manages extended memory, the HMA, and blocks of RAM above 640K but below 1MB (the upper memory blocks, or UMBs).

Loading TSRs and drivers high requires three separate kinds of software working together. First, you must have an XMS driver (such as HIMEM.SYS or a third-party memory manager) to control the UMBs into which the software will be loaded. Second, you must have EMS 4.0 hardware or software to put the memory into the area between 640K and 1MB. (A memory manager such as Qualitas Inc.'s 386MAX is fine on a 386 or on a 286 with Chips and Technologies' NeAT chip set; for other 286s, you need an EMS 4.0 memory board or All Computer's All Chargecard.) Finally, you must have software that supplies a command to load programs high.

DOS 5.0 has two built-in commands to load programs high: DEVICEHIGH (used in CONFIG.SYS for device drivers) and LOADHIGH (LH) for memory-resident programs. To use them, you need to include a line such as:

```
DOS=UMB
```

or

```
DOS=HIGH,UMB
```

to load DOS into the HMA. But if you use DOS's built-in commands, you won't have very much control over the order in which programs are packed into high memory, and you may run out of room. On the other hand, if you don't include a command containing the word UMB but do use a utility such as QEMM or 386MAX to load programs high, you'll be able to optimize the use of your UMBs. DR DOS also comes with utilities that let you load programs high—even on some machines with 286 processors. However, like MS-DOS, DR DOS doesn't come with an optimizer.

Accommodating DOS extenders usually is easy because they bend over backward to work—whether or not you've installed a memory manager on your system— and they have special provisions to work within Desqview or Windows 3.0. As a result, using programs that incorporate DOS extenders isn't a problem on most systems.

Most memory managers, including EMM386.SYS and EMM386.EXE (which come with Windows 3.0 and DOS 5.0, respectively), incorporate the virtual control-program interface, or VCPI, which lets extended DOS programs coexist with memory managers and with Desqview. However, a few programs written with DOS extenders now require your system to have something called DPMI—the DOS protected-mode interface.

DPMI originally was created as part of the memory manager in Windows 3.0—as a way to let Windows programs running in protected mode access DOS, the BIOS, and other low-level portions of the system. However, some DOS programs that use DOS extenders cannot run without DPMI. This means that you must run these programs inside Windows' MS-DOS window (which gives them less memory to work with), or you can enlist the help of a memory manager that supports DPMI. So far, the only memory manager that does this is 386MAX, but the other memory managers on the market undoubtedly will have this capability soon.

Choosing a Third-Party Memory Manager

The best way to organize your memory is to eschew the memory managers that come with your operating system. But with nearly a dozen third-party memory managers on the market, how do you choose?

QEMM and 386MAX are at the head of the pack, and each offers a few distinct advantages. QEMM, for example, works best for Desqview and offers a Stealth feature that sometimes can give you more room in which to load TSRs and drivers high. It also has an excellent analysis tool called Manifest, which lets you explore your system and optimize it for the best possible use of memory. 386MAX is the only memory manager that thus far supports DPMI, and the special PS/2 version of 386MAX —called BlueMAX—has extra features for IBM machines. (Both versions of 386MAX come with ASQ, an analysis utility that also tutors you about PC internals.)

V Communications' Memory Commander is unique in that it can give some programs up to 920K of conventional memory, which is especially handy if your favorite application doesn't support expanded or extended memory. Most of the other memory managers offer some combination of the features sported by these three.

Finally, a company called RYBS has released an optimizing utility, AtLast, that is designed to optimize your system configuration no matter whether you use DOS 5.0, DR DOS, QEMM386, or any other product with memory management capabilities.

Don't Despair

If all of this sounds unbearably complicated, don't despair. The technical support people at companies that sell memory managers sympathize with confused callers and can help you sort things out.

My advice is quite simple: Purchase a brand-name manager that is known to work on your machine, back up the files on your disk (just to make sure), and try running the installation program—which usually will make the right decisions for your system automatically. If this approach doesn't work, don't be timid about calling the product's manufacturer and asking for help.

Your machine undoubtedly will soon be running with more than 600K of free conventional memory, and all your memory-resident programs and device drivers will be tucked safely into high memory, where they won't steal valuable RAM from your DOS programs. ■

Computer consultant Brett Glass is a columnist for InfoWorld, *a weekly magazine covering the personal computing industry.*

Product Information

386MAX, $99.95
Qualitas Inc.
7101 Wisconsin Ave.
Suite 1386
Bethesda, MD 20814
800-733-1377

All Chargecard, $199
All Computers Inc.
1220 Young St., Second Floor
Toronto, Ontario M4T1W1
416-960-0111

AtLast, $79.95
RYBS Electronics Inc.
2590 Central Ave.
Boulder, CO 80301
303-444-6073

Desqview 2.4, $129.95
QEMM386 6.0, $99.95
Quarterdeck Office Systems
150 Pico Blvd.
Santa Monica, CA 90405
310-392-9851

DR DOS 6.0, $99
Digital Research Inc.
70 Garden Court
Monterey, CA 93940
800-274-4374

Memory Commander 2.1, $99.95
V Communications Inc.
4320 Stevens Creek Blvd.,
Suite 275
San Jose, CA 95129
800-648-8266
408-296-4224

Questions & Answers

MEMORY MANAGEMENT

Q Can you explain the difference between extended and expanded memory?

A Extended memory and expanded memory were devised to get around the limitations of the original IBM Personal Computer, which had an absolute memory ceiling of 640K. The scheme for expanded memory, often called EMS, was developed first; it works on any PC, regardless of which processor chip the machine uses. Expanded memory works by switching a block of memory in and out of an area where your computer can see it—the equivalent of taking one piece of paper off your desk and putting another one on it. Switching information in and out of the address space takes time, but for most early PCs this was the only way to circumvent the 640K memory barrier.

Then the IBM PC/AT, with its 80286 processor, arrived on the scene. It has the ability to use more than 1MB of memory and to "see" the additional memory all at one time rather than shuffling it in and out of view. This type of memory arrangement is called extended memory. Extended memory didn't catch on right away because so few of the machines in use could take advantage of it. But now that 8088 and 8086 machines have lost market share to 286s, 386s, and 486s, extended memory has become the preferred way of adding more than 1MB of memory to a computer system.

Nowadays, if you have one of these machines and run a program that expects expanded memory, DOS 5.0's EMM386.EXE—or any other 386 memory manager—will turn extended memory into expanded memory for the benefit of that program. If you have a 286, read on; the next question and answer discuss ways to get expanded memory on your machine.

Q I have a 286 computer with 2MB of extended memory. How can I configure my extended memory as expanded memory? I have some programs that need expanded memory.

A To make extended memory look like expanded memory, you'll need a program called an EMS emulator, or LIMulator. (The term is a pun on the names of the three companies that first endorsed the Expanded Memory Specification: Lotus, Intel, and Microsoft.) Several magazines and shareware authors have made 286 LIMulators available for downloading from BBSes and on-line services; you also can find them on the shareware disks advertised in the backs of computer magazines. If you'd rather buy a shrink-wrapped commercial product, check out your nearest software store.

If you need expanded memory and want to load TSRs above 640K, you should consider other products. If you have a system with the Chips and Technologies' NeAT or LeAP chip set, Qualitas' 386MAX 6.0, or Quarterdeck Office Systems' QRAM, it will not only emulate expanded memory but will allow you to load TSRs into the area above 640K. To see if you have this chip set, open up your computer and examine your machine's motherboard. If you see a chip bearing the word *CHIPS* in large block letters and the number 82C212 underneath, these products will work well for you. If you don't have that chip on your motherboard, you can purchase a hardware accessory called the All Chargecard that will give you the same capabilities.

Q I've just installed DOS 5.0 on my system. How do I interpret the output of its MEM command?

A When you invoke MEM without adding command-line switches, it displays some simple statistics about your system's memory. Those statistics will look something like those in Figure 1.

The first section of the output deals with conventional memory—that is, the memory below the 1MB mark. The total conventional memory is the amount of RAM your system has below the 1MB boundary. It's nearly always 640K, unless you've run a utili-

MEM Command Statistics

```
655360 bytes total conventional memory
655360 bytes available to MS-DOS
403504 largest executable program size

3588096 bytes total EMS memory
1867776 bytes free EMS memory

3145728 bytes total contiguous extended memory
0 bytes available contiguous extended memory
1867776 bytes available XMS memory
MS-DOS resident in High Memory Area
```

ty such as VIDRAM to expand it. The amount of memory that's available to DOS usually is the same number; it decreases if you load a TSR or device driver that installs itself at the high end of conventional memory. The largest executable program size gives the maximum amount of conventional memory available to run a program. (In the sample display above, less than 400K is available—a sign that you should be thinking about loading TSRs and device drivers high.)

The next two lines cover expanded memory. The total EMS memory and free EMS memory tell you how much expanded memory you have and how much is left, the figure for total contiguous extended memory gives the amount of extended memory in your machine, and the following line indicates that no contiguous extended memory is available for use by your system. In this case, that isn't cause for concern. No memory is available because it's being managed by the XMS driver (Microsoft's HIMEM.SYS or a program that replaces it). The amount of available XMS memory is a better guide to the amount of extended memory that's available.

On some machines, you may notice that the total EMS memory and the total contiguous extended memory add up to more RAM than you have in your machine. It would be nice if this meant that more RAM had magically appeared, but that is not the case. It indicates that you're running a memory manager, such as 386MAX or QEMM, that makes the same memory chips available for use as expanded or extended memory, depending on what your programs need at the moment. In this case, the larger of the two numbers should be taken as the amount of RAM your system has above 1MB.

Q What do each of the switches associated with the MEM command do?

A MEM accepts three switches: /C, /P, and /D. The /P (for *program*) switch shows a map of all the programs that DOS has loaded into memory. It's useful if you want to see what device drivers and TSRs you've loaded and how much room they occupy. If, however, you're using a third-party memory manager to load programs into high memory, this switch won't tell you about them; you'll need to use that memory manager's status utility to receive a full report.

The /D (for *debug*) switch displays even more information about memory, including lots of information on internal device drivers and XMS. You should use it only if you're interested in the inner workings of DOS.

The /C (for *classify*) switch classifies programs according to where they're loaded and shows how much memory they use. It also lists the size of programs, provides a summary of memory in use, and gives the size of the largest available block of memory. Like the /P switch, however, it doesn't show programs that have been loaded high by a third-party memory manager.

Q I have 386SX machine and am trying to determine the best use of my 2MB of extended memory. How much memory should I devote to such things as a disk cache and a RAM disk to make certain that I get the best overall performance from my system?

A If you want Windows to run in 386 extended mode (which requires 1MB of free XMS memory), you shouldn't commit more than 1MB of extended memory to TSR programs. Also, most disk caches don't work well unless they have at least 512K of memory available (1MB is preferable). So, start with a 1MB disk cache and deduct the size of the RAM disk you need, but don't exceed 512K because your disk cache needs the other 512K. If this arrangement cramps your style, consider buying more RAM.

Q When a device driver such as RAMDRIVE.SYS or SMARTDRV.SYS has the option of loading into either expanded or extended memory, which one should I use?

A Use extended memory. TSRs and device drivers that use expanded memory sometimes make Windows unstable.

Q DOS 5.0 includes two memory managers, HIMEM.SYS and EMM386.EXE. What is the difference between the two? Do I always have to use them both?

A HIMEM.SYS is an XMS driver, a driver that supports the Extended Memory Specification. It runs on 286, 386, and 486 machines and provides a means of loading DOS into the high memory area (HMA) to free up conventional memory. EMM386.EXE is a 386 memory manager that you can load on top of HIMEM.SYS if you have a 386 or 486 machine. It gives DOS 5.0 the ability to load TSRs and device drivers into high memory, freeing up still more conventional memory.

Third-party memory managers—V Communications' Memory Commander, Quarterdeck Office System's QEMM386, and Qualitas' 386MAX—do everything that HIMEM.SYS and EMM386.SYS do, only faster and better. So, if you buy one of these products, you can consign DOS 5.0's memory managers to the scrap heap.

Q When I installed DOS 5.0, I was looking forward to loading device drivers into high memory. However, even though the installation procedure placed HIMEM.SYS in my CONFIG.SYS file, the MEM /C command shows no high memory available. Did I do something wrong?

A Yes. As mentioned above, you need to load the EMM386.SYS device driver to load programs high—assuming, of course, that you have a 386 or 486. If you have a 286, you'll need 386MAX or QRAM if your computer has a NeAT chip set, and an All Chargecard if it doesn't.

Q When I attempt to use DOS 5.0's DEVICEHIGH command to load a device driver into upper memory, it loads into conventional memory instead. This occurs even though the MEM /C command reports that the amount of memory available in upper memory is greater than the amount of memory required by the device driver. Why doesn't the driver load into upper memory?

A It may be that your upper memory area is fragmented. For a driver to load high, it needs to fit into a single, contiguous upper memory block (UMB). Use a third-party memory manager or optimization program to optimize upper memory usage.

Q What factors determine the number and size of UMBs available for loading programs high?

A If your system has lots of peripherals that map their RAM or ROM into high memory, they may take up a great deal of space—and fragment what you have left. If this is a problem, and you have a 386 or 486, try QEMM; its Stealth feature sometimes can get around this problem by mapping the peripherals in and out of memory on the fly.

Q I recently installed some hard disk compression software on my 286 system. The software works fine, but now some of my programs won't run because too little memory remains. How can I increase the available memory for running programs on this system? I have 1MB of RAM.

A Many compression programs can load themselves into high memory—and store some of their data in expanded or extended memory. Your program's manual will tell you what your particular program is capable of doing. Most systems are designed to turn the "extra" 384K of memory (that is, the memory left over after you deduct 640K for conventional memory) into extended or expanded memory without relying on additional software. If your program can do this, by all means take advantage of this feature. (Unfortunately, on some 1MB systems, this memory goes to waste, and you need to add more RAM to get extended or expanded memory.) If you find that you have the wrong kind of memory to work with your compression program (say, extended instead of expanded), or want to load the compression program high to free up space, refer to the second question and answer above. ■

Tip 5

Into the Void

Most users familiar with batch files know all about the ECHO statement. Typing:

```
ECHO OFF
```

provides a means of keeping DOS from displaying (echoing) command lines to the screen as they execute. The ECHO statement itself still will appear on the screen, unless you have DOS 3.3 and later and preface the command with an "at" symbol (@):

```
@ECHO OFF
```

Unfortunately, though, the command isn't foolproof. Many programs (including most DOS utilities) simply turn echoing back on when they start, or bypass the ECHO function in other ways. However, most DOS programs and utilities are designed to honor DOS device redirection—the NUL device among them.

NUL serves as a data trash can of sorts; many DOS programmers refer to it affectionately as "the bit bucket." Dumping unwanted messages into this bit bucket from within a batch file is a simple matter of adding the greater than (>) redirection symbol followed by the word *NUL*. Adding the following line to your batch file, for example:

```
COPY C:\PW\TEXT1.TXT A:\ > NUL
```

prevents the message "1 file(s) copied" from appearing on the screen after TEXT1.TXT is copied from one directory to the other.

NUL also can help out if you want to type at a keyboard without fear that DOS will interpret what you type as a command. When you type the following at the DOS prompt:

```
COPY CON > NUL
```

DOS redirects to the NUL device anything that you subsequently type, instead of passing it along to the DOS command processor for execution. To cancel device redirection, press the F6 key and then press Enter.

HARDWARE

On The Upgrade Path

Understanding DOS can improve your efficiency, but sometimes what you need to give your computing a real boost is a change of hardware.

by John Wolfskill

In 1988, a high-performance 80386 computer came with a $6,000 price tag, and an entry-level system set you back a cool $4,000. Third-party hardware vendors were quick to offer alternatives— CPU accelerator cards, faster motherboards, and the like—to those who wanted current technology without the steep price tag.

These days, 386 and 486 systems sell for less than half of what comparable models cost in 1988, and competition within the PC industry has lowered the price of component hardware. That's good news for those in the market for a new PC. In fact, now may be the ideal time to trade your old 8088 or 286 machine for a fast 386. For many PC users, however, a RAM upgrade or more disk storage space, may be all that's required to keep yesterday's computer up to speed with today's sophisticated applications.

Is it wise to add performance Band-Aids to your old PC, or better to buy a new system? The answer depends on the equipment you now own and the level of performance you expect. This article will help you make an informed choice, based on the current state of PC hardware and your computing needs. All prices mentioned reflect what you'll pay today for products sold by reputable businesses specializing in upgrade hardware.

Giving Your 8088 a Leg Up

If your PC is equipped with less than 640K of memory, a memory upgrade is the first upgrade option you ought to consider. It's the easiest upgrade to perform, and it typically costs less than $75. RAM chips are available from a number of mail-order specialty houses, or from your local PC dealer.

The best way to make sure your new chips will prove compatible with your existing RAM chips is to open your computer's case, locate the chips (look for several rows of 14-pin chips arranged in groups of eight or nine), then jot down the numbers that appear on top of each chip. Armed with this information, you can help your dealer match the specifications of your new RAM chips to those for the old ones. An 8086- or 8088-based system that operates at 10MHz or slower requires chips with a rated access time of 150 nanoseconds (ns) to 200 ns. Faster chips will cost more but will not provide better performance.

The second item on your upgrade list probably should be a expanded memory card. An 8088-class computer can address only 1MB of conventional memory directly; installing a so-called EMS card in an empty expansion slot inside your computer makes it possible for your system to work with more than 1MB of data at once. A typical LIM EMS 4.0 expansion board with 2MB of RAM costs between $200 and $225. Spreadsheet programs such as Lotus 1-2-3, Borland International's Quattro Pro, graphics applications, desktop publishing programs, and most programs designed for

Microsoft Windows 3.0 will use expanded memory automatically when it is available.

If your 8086 or 8088 system contains a single floppy drive or even two floppy drives, a hard drive should come next on your upgrade agenda. Although you'll find many types from which to choose, the new integrated drive electronics (IDE) models are reliable and cost-effective solutions for systems in the 8086 and 8088 category. With these drives, the disk controller circuitry is an integral part of the drive, a setup that helps to eliminate the compatibility problems you may encounter if you purchase a drive and drive controller separately.

Most IDE hard drives come with a connecting cable, so the only extra item you must buy is an inexpensive ($20 to $25) IDE host adapter that plugs into an expansion slot and provides the IDE circuitry necessary for your CPU and the new drive to work together. An IDE hard drive will provide a substantial boost in your computer's performance at a moderate cost. Moreover, it will make your computer more convenient to use. A typical 80MB IDE hard drive costs $250 to $300 from mail-order houses—slightly more than that if you purchase it through your local computer dealer.

Using an IDE drive with an older PC has one potential drawback, however. On some systems (typically those manufactured before 1990), the drive parameters—information such as the number of heads and so on— are burned into the BIOS, the chips containing the set of instructions your PC reads and executes when you first turn on the power switch. This causes compatibility problems if your old drive parameters don't match the ones for your new drive. Be sure to check with your computer dealer or manufacturer to make certain your system's BIOS is compatible with the new IDE drive before you purchase it.

Another high priority should be upgrading a 360K floppy disk system to the high-density drives that use 1.2MB 5¼-inch or 1.44MB 3½-inch disks. Many software distributors now provide their applications exclusively on high-density disks. If the ability to share files with other users is important to you, consider a high-density floppy disk system a necessity.

If you're already upgrading to an IDE hard drive, a high-density floppy disk upgrade isn't expensive. A combination IDE host adapter/floppy disk controller card costs about $70, for example, while a 1.2MB and 1.44MB floppy drive system, with connecting cables, costs about $80 to $90 for each drive.

If your 8086/8088 operates with a standard CPU clock (4.77MHz or 8MHz), increasing the internal clock speed of the CPU or replacing the CPU with a faster model also will increase your computer's overall performance. You need a moderately fast CPU (16MHz to 20MHz) if you plan to run applications in graphical environments such as Windows or Digital Research's GEM , or you routinely need to calculate large spreadsheets.

Unfortunately, the once expansive CPU accelerator card market has dwindled to a few players and a few choices. Sota Technologies sells a reliable coprocessor accelerator card called the 386si. This half-length PC expansion card comes with a 16MHz 80386 CPU that will provide a substantial performance boost for a modest outlay ($400 retail). The 10MHz Sota 286i ($280) provides a smaller speed boost in the form of an 80286 CPU mounted on the same accelerator card used for the 386si.

The final step in "modernizing" your 8086/8088 system is a video upgrade. For most users with 8088-class machines, this upgrade path involves investing in both a new video display card and a monitor. Unless you have special needs, the choice is a simple one: Get a standard VGA video adapter card and a matching color monitor to replace your existing video components. Many computer dealers and mail-order houses now take the worry out of buying the correct components by offering standard VGA video adapters and monitors as sets.

A new video system once was considered a dauntingly expensive upgrade. Fortunately, the prices of VGA systems are falling right alongside the prices of new computers. Many mail-order sources offer a name-brand VGA adapter that can produce 640 horizontal dots by 480 vertical lines and 16 colors for as little as $65. A standard 14-inch VGA analog monitor sells for $200 to $300.

Almost all VGA cards are designed to fit in a 286- or 386-class computer. They come standard with a 16-bit data bus, which carries data from your computer to the video card, but 8086/8088 owners can use the card in their computers' standard 8-bit expansion slots. That's because most VGA cards include bus-detection circuitry that senses whether an 8- or 16-bit is in use and adjusts the card's internal circuitry and bus timing to match the computer's.

Upgrading Your 286 System

A memory upgrade also is the logical starting point for upgrading a 286-based system. The original IBM PC/AT came with a 6MHz 80286 central processor, a 30MB hard drive, and 512K of RAM. Later model AT compatibles had CPU clock speeds ranging from 8MHz to 20MHz; system board memory ranged from 640K to 4MB.

If yours is an original IBM AT or compatible (such as the Tandy 3000HL) and it has just 512K of memory, a cost-effective way to increase your computer's memory to 640K is with a 128K memory "top-off" board. Boca Research Inc. sells a 128K board ($145) that will do the job. Tandy Electronics National Parts also sells a memory upgrade kit ($59.95) that lets you upgrade the 3000HL or 3000NL to 640K.

Unlike 8088-class machines, 286s can address extended memory directly, and a number of companies offer reasonably priced extended memory boards that will do the trick. For example, JDR Microdevices Inc. sells its 8MB Simmply RAM card for $199.95. This expansion card lets you designate the board's memory as either extended or expanded (see "Sorting Out Memory" on page 38 for a discussion of the distinctions). It can accept either 256K or 1MB single in-line memory module chips, chips designed to be placed on a circuitboard in combination with a group of other similar chips.

Adding a hard disk with higher capacity is relatively simple for 286 owners. Almost all 286s came with a drive controller that handled both a high-density floppy and a hard drive. Before you purchase a larger or faster hard drive to replace your original (or before you add a second hard drive), you need to know the type of hard disk controller installed in your computer. As with memory chips, the best way to make sure your new hard drive is compatible with your existing drive controller is to open the case, locate the controller card, then jot down the manufacturer's make and model information. Your dealer can match the specifications of your new hard drive to your existing drive controller.

One look at the shelves in your local software store will reveal that Microsoft Windows is becoming the platform of choice for mainstream PC business applications. If you plan to use your computer with Windows 3.0, a fast 16-bit super VGA video card is as important as a speedy hard drive. For $299, you'll find a fast and reliable super VGA card in Diamond Computer Systems' SpeedStar VGA. The board comes with 1MB of display memory, the amount of memory you need to achieve a resolution of 1024 by 768 pixels in 256 colors.

If your present 286 system isn't already equipped for VGA graphics, you'll also need to purchase a new monitor. To make the most of a super VGA graphics card, you need a super VGA analog monitor or a so-called multifrequency monitor that can display 1024-by-768-pixel VGA graphics screens. A popular multifrequency model is NEC's 14-inch Multisync 3DS color monitor ($699, retail). It's a good choice for a wide variety of super VGA applications.

Most 286s with a CPU that runs at 10MHz or faster can cope with graphics-based applications, such as those designed for Windows, without the need for a CPU accelerator. However, 6MHz or 8MHz models will benefit from a coprocessor accelerator such as Intel's Inboard 386 ($799). The kit comes with a full-length 16-bit expansion card, which contains a 16MHz 80386 CPU and 1MB of 32-bit RAM (a 2MB memory daughtercard also is available for $495). Although this solution costs more than an accelerator card that uses your computer's existing memory, it provides substantially better performance. A word of warning, however. The kit is moderately difficult to install. If you've never tackled an upgrade that requires yanking a firmly embedded 80286 CPU chip from your computer's motherboard, you may be better off assigning the task to your local computer technician.

Examining All the Options

Of course, there are many more options in each category that you can and should consider if you have special needs. But be aware that a complete hardware overhaul won't necessarily cost you less than buying a new system. In fact, in today's PC market the total cost of upgrading a stock 8088- or 286-class system to minimal 386 performance levels typically costs several hundred dollars more than purchasing a complete system from a reliable dealer or mail-order house. Still, upgrading a few carefully selected components can breathe life into your 8086, 8088, or 286 system, and allow it to grow old gracefully. ■

John Wolfskill is a free-lance writer and president of Powertrain Development, a New Hampshire-based software development company.

Product Information

TopHAT (128K), $145
Boca Research Inc.
6413 Congress Ave.
Boca Raton, FL 33487
407-997-6227

128K Memory Upgrade (25-4082), $59.95
Tandy Electronics National Parts
900 E. Northside Dr.
Fort Worth, TX 76102
800-442-2425

386si Card, $400
Sota Technologies
559 Weddell Dr.
Sunnyvale, CA 94089
800-933-7682

Inboard 386 PC, $799
Inboard 386 Piggyback (2MB), $495)
Intel Corp.
Personal Computer
Enhancement Operation
5200 Elam Young Parkway
Hillsboro, OR 97124
800-538-3373

Multisync 3DS, $699
NEC
1414 Massachusetts Ave.
Boxborough, MA 01719
508-364-3000

Simmply RAM, $199.95
JDR Microdevices
2233 Samaratin Dr.
San Jose, CA 95124
800-538-5000

SpeedStar Hi-Color (1MB), $299
Diamond Computer Systems Inc.
532 Mercury Dr.
Sunnyvale, CA 94086
408-736-2000

Questions & Answers

HARDWARE

Q What are the differences between a serial mouse and a bus mouse? Is there any reason to prefer one type over the other?

A The only real difference between the two types is the data path each uses to relay information from the mouse pointer to your application programs. A serial mouse attaches to one of your computer's RS-232 serial ports, while a bus mouse attaches to an adapter card you install in an empty slot in your PC's expansion bus.

Neither type offers a performance advantage. Choose the type that is more convenient for your system. If, for example, your system has a pair of serial ports, you can operate a modem and serial mouse without using a precious expansion slot. Some PCs, such as the IBM PS/2, provide a bus-type mouse port directly on the computer's system board, which eliminates the need for an expansion card.

Q When my computer will be unused for a period of time, is it better to leave it running or to turn it off?

A Turn it off. It's a common misconception that turning your computer's power switch on and off several times a day will subject it to a great deal of electrical stress. Modern computer power supplies are designed to handle the normal AC voltage and current fluctuations that occur when you operate a computer's power switch. (Try flipping the switch on and off rapidly. You'll find the computer will refuse to start, and its power supply will simply ignore the rapid flicks of the switch.)

Not only is it safe to turn your computer off when you won't need it for a few hours, but it's highly advisable. In addition to lowering your electric bill, shutting off your system will benefit your PC's internal components. If your PC has a hard disk drive, its drive motor spindle operates at 3,600 revolutions per minute. Each minute that the drive operates brings its mechanical components that much closer to inevitable failure from heat and friction.

Any PC eventually will wear out due to a variety of physical and environmental factors. Leaving your system on when you don't need it only accelerates the process.

Q I own a Ricoh AT-compatible PC, which was manufactured in 1986. It worked fine until my boyfriend stored it away for the summer. Now, the hard drive seems confused. I can get a C> prompt and display a directory of files. However, some of the directory entries appear jumbled. Worse, the 30MB drive now thinks it's a 20MB drive. Will I need a new hard drive?

A You can try several things before you lay out money for a new hard drive. The problem may be in your computer's setup program. Each time you turn on your computer, this program refreshes your PC's memory about the type of hard drive, monitor, keyboard, and other components that are installed in your system.

A setup program usually is stored in a PC's RAM. The program is kept alive by AC power when you operate the computer and is maintained by an internal DC battery when the machine is turned off. Some computer manufacturers' setup programs install default settings when the internal battery dies. If that happens, your computer looks at the default setup data, which may include a 20MB drive, a common type of hard drive in 1986. You can set problems of this sort right simply by replacing the battery.

To do this, slide the cover off your PC and substitute fresh batteries for the old ones. The batteries can range from small, flat, 3-volt lithium cells, such as those found in wristwatches, to a series of 1.5-volt AA alkaline cells. If you can't locate anything that looks like a removable battery, your system may include a rechargeable battery (it looks like a small oil drum) that's permanently soldered to the system board. If you're not handy with a soldering iron, ask your local PC technician to remove it, test it, and substitute a fresh battery if that is necessary.

After installing a fresh battery, run your system's setup program and provide the appropriate information for all of your hardware devices, including the hard drive. If the setup lists the drive only by type—by calling it type 7, for instance—refer to your hard drive manual, or call the drive manufacturer to determine the number. Hopefully, this will take care of your problem.

Q I have a Leading Edge XT compatible (circa 1985) in which I recently installed a new 40MB hard drive and disk controller card.

After installing the drive and controller, I changed the setup program to include the new hard drive. All I received for my efforts was the error message, "Drive not recognized." What's the problem?

A First, check your drive's controller card and cables. The controller card should be firmly seated in the expansion slot; the cables should connect the drive to the controller card and be firmly attached to the power supply. Installing a controller card cable can be tricky, because it's possible to plug the cable into the controller card in reverse. If you look carefully at the thin ribbon cable, you'll notice a narrow red strip on one side. This end of the cable normally attaches to pin 1 of the hard drive and disk controller interfaces. However, read the instructions for your controller card carefully to make sure this is the case with your drive.

If you're satisfied that the cables are installed correctly, and you still get no response, you may have a BIOS problem. If your Leading Edge PC still has its original 1985 BIOS, it may not include the proper internal drive parameter tables to allow the computer to successfully install the 40MB hard drive.

If this is the source of the problem, you have at least two possible solutions. First, you can remove and upgrade your system's BIOS chips. Call your local Leading Edge dealer to obtain the latest version of the Leading Edge BIOS (the chips usually cost $20 or less). Or you might try one of several software utilities from Ontrack Computer Systems (Eden Prairie, MN, 612-937-1107). This memory-resident utility, which you install in your computer's CONFIG.SYS file, may enable it to recognize the hard drive.

Q I recently purchased a new 386SX computer, complete with a super VGA monitor, 40MB SCSI hard drive, 3MB of RAM, dual floppy drives, and DOS 5.0. It ran Windows 3.0 fine until I made the computer part of our office network. Now the PC locks up whenever I start Windows. What gives?

A The problem is most likely a memory conflict between your network software and Windows. When you load Windows, the two programs collide because they're attempting to share the same memory space.

To resolve the difficulty, first try modifying the Windows SYSTEM.INI file, to reserve a portion of RAM for the exclusive use of the network code. The EMMEXCLUDE command is provided for this purpose. Ask your network administrator which RAM segment address your network uses to install its operating code in memory. The segment usually is identified by a range consisting of two four-digit hexadecimal numbers, such as C000–CFFF. Load the SYSTEM.INI file into your word processor, then find the [386enh] section of the file, and add the following line:

```
EMMEXCLUDE = xxxx:xxxx
```

where *xxxx:xxxx* is the memory segment range obtained from your network administrator.

An alternative is to ask your network administrator's help in relocating the address of the network code—by using software commands or resetting a jumper switch on your system's network card. This approach lets you resolve the conflict by selecting a different memory segment for the network operating code instead of preventing Windows from using the memory area occupied by the network.

If none of these approaches works, perhaps your network's memory buffer requirements for direct-memory-access (DMA) data transfers are larger than what Windows provides as a default setting (Windows' default allotment is 0K for a 286 system). To reset the DMA buffer size, load Windows' SYSTEM.INI configuration file into your word processor, then locate the section of the file labeled [386enh]. To reserve a 32K DMA buffer, add the following line:

```
NETDMASIZE=32
```

Save the modified SYSTEM.INI file, then restart Windows.

Another potential problem is that Windows' data-transfer buffer is smaller than what your network software needs. You can resolve this problem by adding another line to the [386enh] section of the SYSTEM.INI file:

```
NETHEAPSIZE = 16
```

This increases the size of the Windows data-transfer buffer to 16K (the default is 12K). Both of the above examples assume you're running Windows in 386 enhanced mode. If you're running in standard mode, make the same changes to the [standard] section of the SYSTEM.INI file.

If Windows doesn't seem to be the problem, you may want to turn a reprobatory eye toward your SCSI drive, using the same memory segment relocation technique. Your SCSI host adapter may be causing a similar memory conflict. Again, try to relocate the RAM addresses the SCSI driver uses to install its operating code.

Q What is shadow RAM? I am considering purchasing a new computer or at least upgrading mine. Some systems come with shadow RAM, while others come with just plain RAM. As a PC user, what do I need to know about shadow RAM ?

A Shadow RAM is a bank-switching technique commonly used in 286 and 386 computers to allow RAM and ROM to share the same CPU address lines. RAM shadowing allows the operating system (and some specially designed applications) to use the area above 640K in a 1MB system board for storing data in a single contiguous block. The system normally uses this area in patchwork fashion to service the computer's vital ROM BIOS, video display memory buffer, and other hardware devices. Almost all modern system boards now use RAM shadowing because it allows DOS to make maximum use of the area of RAM between 640K and 1024K.

All you need to know before purchasing a new system board is whether it offers shadow RAM or not. The computer's setup program lets you change the shadow RAM address locations and the operation of its special circuits. However, unless you're familiar with the system board's logic chip set, it's not wise to experiment with these settings.

Q I periodically get an error message that reads, "RAM Parity Error—SEGMENT *xxxx*", where *xxxx* is a four-digit number that seems to vary each time the problem occurs. What does the message mean, and how can I correct it?

A This error message is one that your computer's built-in ROM BIOS displays when the system detects a loose or bad memory chip. When a memory chip fails, a ROM BIOS interrupt-handling routine (INT2) displays the segment address of the chip's failed bits. Unless you know how, and in what order, your computer's RAM chips are wired to the computer's address bus, the displayed number may not help you locate the failed chip.

If you're queasy about tinkering inside your computer, take your system to a service technician for diagnosis and repair. If not, you may be able to correct the problem yourself. Open the case and locate your computer's RAM chips. Check for obvious problems: Are all the RAM chips seated firmly in their sockets? Are the chips' metal legs inside the proper socket holes? If your system uses single in-line memory modules, are they firmly seated, with the plastic latch guards snapped into a locked position? Do the DIP switch settings for your system board's memory match the amount of RAM installed in the computer? Are the tops of any of the chips discolored (a white or gray powder may indicate a failure from overheating).

If all or part of your system's RAM resides on a memory expansion card, is the board firmly seated in its expansion slot? Is the board's grounding screw or grounding strap firmly attached to the computer's case?

If everything seems OK, you'll need to test your system's RAM chips. The test procedure is straightforward, but time-consuming. First, buy or borrow a chip you know is reliable. (It also must match the ones installed in your system.) Then, check your system's RAM chips one by one, taking out the old chip, replacing it with the test chip, and reassembling the computer. If the problem persists, remove the test chip, replace the chip you removed earlier, and test the next chip. Continue the process until the error message disappears and the computer operates normally.

If you still get an error message after you've tested all the chips, you may have more than one bad RAM chip. You can continue trying to solve the problem by systematically replacing the chips a bank at a time. If your system uses standard dual in-line pin chips, buy a set of nine matching chips (about $13) and replace one nine-chip bank of RAM at a time.

Keep in mind that you can damage RAM chips with static electricity if they're not seated in their circuit-board sockets. Make sure to discharge built-up static in your body by touching a grounded metal surface (a convenient place is the top of your computer's power supply case) before you handle the RAM chips.

Q How important is it to clean the inside of my computer? Should I use a brush or apply a cleaning solution?

A It's a good idea to give the inside of your PC a thorough cleaning annually. The combination of dust, smoke particles, and other contaminants deposited by the air moving through your computer can form a corrosive layer on circuit boards and the metal legs of chips. Over a period of time, this buildup can wreak electronic havoc, causing short circuits and loss of electrical contact at key connection points, such as expansion board sockets and expansion board edge connectors.

Use a vacuum with a soft brush attachment to remove the buildup. The residue most cleaning solutions leave behind can be more harmful than the dirt and dust. If you live in a humid climate, perform this chore at least once every three months.

Q Before installing a new DOS version, I used the FDISK program to partition my hard drive into two logical drives. Now my computer won't boot from the hard drive, although it will boot from a floppy. What did I do wrong?

A Either you created a pair of non-DOS data partitions or you forgot to set the DOS partition as the active partition before you exited FDISK. Run the program again, checking to make sure one of the partitions is named as a primary DOS partition and that the letter A appears under the Status column immediately to its right. The following display shows a typical setup:

```
        Type     Status   Volume   Size
C:1     PRIDOS   A        MSDOS    20MB
D:2     NONDOS                     20MB
```

Boosting Your PC's Memory

Step-by-step instructions for choosing more memory and installing it in your computer.

by David S. Veale

If you follow the computer press, you're aware of the ever-increasing RAM requirements of software. Many of today's business applications require at least 640K of memory. Microsoft Windows 3.0 runs—barely—within that memory limit. But you need expanded or extended memory beyond 640K just to load newer versions of 1-2-3 and Dbase IV, and you need a minimum of 4MB of RAM to do any serious work with OS/2.

Even if you don't use high-powered programs, you have reason to upgrade your computer's memory. With DOS 4.x and earlier, a print spooler, and a few of your favorite memory-resident programs loaded into a PC with 640K, you may have only 350K or so left.

To upgrade your system for today's RAM-hungry software, you need a basic understanding of computer memory and the new ways manufacturers package RAM. You also must know how to determine the amount of your PC's memory, whether more can be added, and how to shop for and install additional RAM.

Expanded or Extended?

In IBM-compatible machines with Intel 8086, 8088, 80286, 80386, and 80486 processors, you can use three types of memory: conventional, expanded, and extended. The 8088 chip, which powered the first PC and for which DOS still is written, can address as much as 1MB of RAM. When designing the PC, engineers set aside 128K for video memory and 256K for ROM, leaving 640K for DOS, programs, and program-generated data. That 640K is called conventional memory.

Some DOS programs can address more than 640K by using expanded memory, which requires a special board and software for 8086, 8088, and 286 systems and a memory manager for 386 and 486 systems. The hardware and software must support the Lotus/Intel/Microsoft Expanded Memory Specification (LIM EMS) 4.0 standard, which lets some programs use up to 32MB of RAM below the 1MB line by swapping information in and out of a 64K window, or page frame.

Extended memory, available only on systems with a 286 or later processor, is simply an extension of RAM beyond 1MB. It's inaccessible to DOS unless you use software with special DOS-extender coding to switch the processor between real mode, used by DOS, and protected mode, which can access memory beyond 1MB. A few programs switch the processor into protected mode after they're loaded. But until more such software appears, extended memory will mainly interest users of Windows 3.0 and high-powered operating systems such as OS/2 or Unix.

If you have a 8088- or 286-class system and programs that use expanded memory, don't upgrade your system board's RAM beyond 640K; you need an expanded memory card. But if you have a 286 and want to use a disk-caching program that can take advantage of extended memory, add that kind of memory. You may need a mix of memory types.

How Much RAM Do You Have?

When you turn on your computer, it probably will run a memory test just before DOS loads; if it does this, it will display on screen the amount of memory installed. If it doesn't, type:

```
CHKDSK
```

at the DOS prompt to learn how much conventional RAM your machine has and how much of each type is free. Utility programs such as Central Point Software's PC Tools and Symantec Corp.'s The Norton Utilities also will give you this information. DOS 4.0 or later comes with a utility called MEM.COM that will tell you the total amount of memory installed in your system and how much of each type is free.

If you own a system with an 8086 or 8088 processor and haven't boosted DOS's addressable RAM with a memory expansion card and software, the most memory your system will have (besides that assigned to video memory and ROM) is 640K. If you have a 286 or later system, the power-on memory test still is your best source of information on total conventional and extended memory.

Remember that expanded memory won't be counted in the memory test. An expanded memory card always has an accompanying software driver, which often displays the amount of expanded memory available when it loads. If you have a 386 or later system, you may have a memory manager that makes extended memory emulate expanded memory. Note that the amount of memory used this way is subtracted from the total extended memory available.

You can verify the accuracy of the memory count message by looking inside your computer. Make sure that the place in which you'll work is static-free and that your system and all attached peripherals are turned off or, better yet, unplugged. Memory resides on the system board and on expansion cards. After you remove your computer's cover, you may have to remove expansion cards to reach your system board. Most newer 8088- or 8086-based systems have 640K (or room for it) on the system board; older systems allow as little as 64K to 256K on the system board and rely on expansion cards for the rest. Others, especially 386s and later, may have all of their memory on expansion cards. Check your system's documentation for details.

To determine if your computer uses conventional RAM chips (approximately 3/4- by 1/4-inch chips), look for rows of similar chips. The group of digits 1256 within the number indicates that the chip is one bit wide and contains a column of 256,144 bits (256 kilobits, or Kbits). These chips aren't useful until they're arranged into 8-bit units, or bytes.

To understand the step from bits to bytes, think of the memory in one of these chips as a small rectangle one bit wide and 256Kbits high. Next, imagine arranging eight of these chips in a row eight bits wide. The row would contain 256,144 bytes, or 256K of RAM.

Now, mentally add another 1-bit by 256-Kbit chip to the row. The ninth chip provides what is called a parity bit. All IBM systems and most compatibles work on an even-parity system. A bit may have one of two values, zero or 1. When a byte is stored in memory, the computer counts the number of 1's in the byte. The parity bit then is set to a zero or a 1 so that the number of 1's in the byte, plus the parity bit, is an even number. For example, this byte:

```
10001010
```

has three 1's. If the byte were stored in memory, the parity bit would be set to 1 because three plus one equals four, an even number. If the byte had two or four 1's, the parity bit would be set to zero because two and four already are even numbers. Every time a byte is read, the number of 1's in it is counted to be sure parity is even. If a bit has changed—for instance, due to a power fluctuation or chip failure—the byte will have an odd number of 1's, and the computer will know that the byte is corrupted. Usually, the computer will display an error message and lock up. (If two bits change, the number of 1's will produce an even number, and the computer won't detect the error.)

A group of memory chips that makes up a set of complete bytes (plus one bit for parity) is known as a memory bank. You may find banks of memory containing various numbers of bytes: 16K or 32K (probably only in older machines), 64K, 256K, or 1MB. Each will be nine bits wide. Usually laid out in straight rows on the system board or memory expansion board, banks are easy to spot.

To determine how your system's RAM is arranged, locate its memory banks and use the chip's code numbers to figure out the number of bytes in each bank. Chips usually are one or four bits wide, which means you should find them in sets of nine to a bank (eight 1-bit chips plus one 1-bit chip for parity) or sets of three (two 4-bit chips plus a 1-bit chip for parity). The diagram on page 54 illustrates these common bank configurations.

My computer has two rows of 1-bit by 256-Kbit memory chips; there are nine chips per row. Each row constitutes a memory bank. The computer looks at eight of the chips in each row (the ninth is for parity only) as one large rectangle 256Kbits high and 8 bits wide, or 256K of RAM per row—a total of 512K of RAM.

Sometimes a row of chips may contain more than one memory bank. For instance, in addition to the two

256K banks I mentioned, my PC also has a row containing four 4-bit by 64-Kbit chips and two 1-bit by 64-Kbit chips. Two of the 4-bit chips plus a 1-bit chip constitute a 9-bit-wide bank of memory containing 64K of RAM. Because the row contains four 4-bit chips and two 1-bit chips, it has 18 bits, or two banks—a total of 128K of RAM. Along with the first two banks of 256K each, this additional 128K makes 640K of total system memory.

Many well-known companies, including IBM, Advanced Micro Devices, Tandy, and Texas Instruments, manufacture chips with the same or similar part numbers.

SIMMs: A Technological Breakthrough

A relatively recent technological advance is the single in-line memory module (SIMM), an entire bank of memory mounted on a circuit board about the size of your thumb. SIMMs come in 256K, 1MB, or 4MB capacities, with a 16MB chip on its way. Each board contains enough tiny chips (usually two 4-bit chips and a 1-bit parity chip or eight 1-bit chips and a 1-bit parity chip) to make the SIMM nine bits wide. One 9-bit wide SIMM constitutes a memory bank.

A SIMM snaps into a specially designed socket on a memory expansion board in much the same way that an expansion card fits into an expansion slot in your PC. Because SIMMs can be mounted perpendicular to the board, they hold more memory in less space than conventional chips. In addition, SIMMs are less prone to failure, have no pins to bend or break, and are easier to install and remove. Many of the major computer manufacturers, including IBM, Compaq, Tandy, and Dell, now use SIMMs almost exclusively for memory. SIMMs may soon replace conventional RAM chips completely.

Checking Memory Speed

The last portion of the part number on a memory chip typically indicates its speed. The number 10, for example, indicates that the chip runs at 100 nanoseconds (ns). Sometimes, as on the chips in my system, only the first two digits of the speed value appear; the part number on a 100 ns chip would contain a 10, but an 80 ns chip might be marked 80. RAM chip speeds usually range from 60 ns to 200 ns.

When adding or replacing RAM, use chips of the same speed as the RAM the manufacturer included in the system. Using faster chips won't make your system run any faster. The speed rating means only that the chip is *capable* of running at a given speed—not that it *will* run at that speed if installed in a system designed to access RAM at a slower rate. Also, because chip performance can vary considerably from a chip of one speed to a chip of another and from one brand to another, mixing chips of different speeds or brands within the same bank can result in scrambled data when the CPU attempts to read the entire bank at one time.

Many memory chips carry markings similar to those in my machine, but standardization isn't absolute. If you encounter chips or SIMMs with no markings, consult a chip reference book, catalog, or supplier for help.

Deciding What to Buy

Once you've identified the amount and type of memory in your system, it's simple to determine if and how you can add more. If your computer uses conventional 16-pin RAM chips, check your system board for groups of empty sockets that would accommodate this type of chip. Many 286-class systems have room on the system board for at least 1MB or 2MB of RAM and often have empty banks of nine sockets each for additional chips.

If your system board is full, check your documentation to see if you can replace 64K chips with 256K chips or 256K chips with 1MB chips. If your board can't be upgraded, consider adding memory boards in empty expansion slots. When shopping for this type of board, be sure it gives you the type of memory you need and is compatible with your system (choose an 8-bit board for an 8088 or 8086 machine and a 16-bit board for a 286). Some 386 and 486 systems require their manufacturers' proprietary 32-bit memory cards.

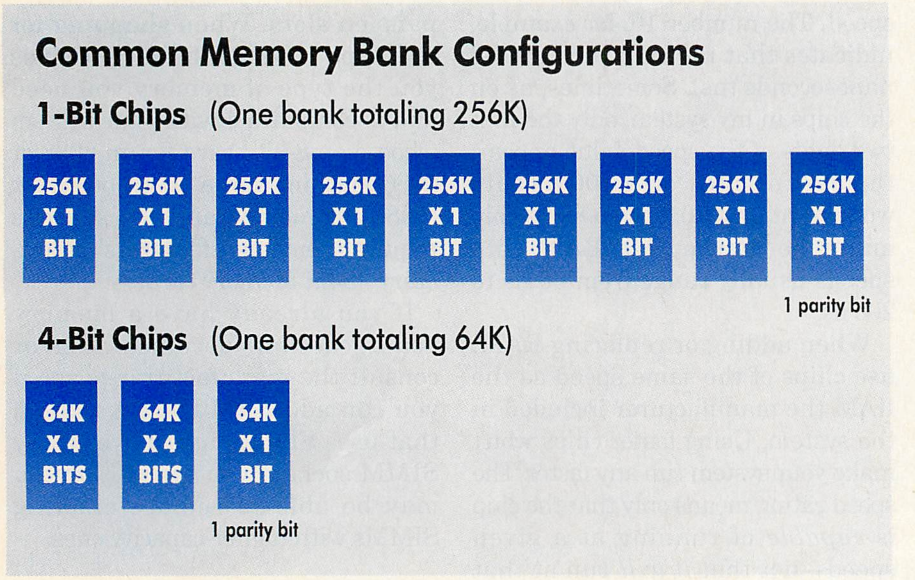
If you already have a memory board, check its documentation or consult the manufacturer to see if you can add RAM to it. A system that uses SIMMs may have empty SIMM sockets you can fill, or you may be able to replace existing SIMMs with higher-capacity ones.

Installing RAM Chips

Before installing new RAM chips or SIMMs, take these precautions: Make sure your body is static-free. (I recommend using a commercial grounding strap, available at electronics stores for about $5.) And be patient; the pins on a chip bend and break easily. When you're ready to start, look for a notch or dot on one end of the first chip. Match that notch or dot to the corresponding dot in one end of a socket. The markings on the chips and sockets must align for the chips to work properly.

If the row of pins on the chip are spread too far apart, they won't fit into the socket. Lay the chip on its side on a hard surface and press down gently to move the pins inward. Once the pins line up with the holes in the socket, hold the chip by its top with your thumb and fingertips and push it down firmly so that each pin goes into a hole. (You can use a chip inserter, which tends to force the pins into the right position for installation and helps you to avoid bending them, but it may prevent you from feeling whether the chip is going in straight.)

Look at the installed chip from both sides. If you discover bent pins, remove the chip. I recommend using a chip puller, but you also can slide the tip of a small, flat-blade screw-



Common Memory Bank Configurations

driver between the chip and its socket, using a gentle rocking motion to lift the chip. Then use your fingers, needle-nose pliers, or a pin straightener to fix the bent pin and try again. A bent pin is fragile; take care not to break it off. Repeat the entire process for each chip.

Installing SIMMs

Because they don't contain pins that can bend or break, SIMMs are easier to install than chips. A SIMM sits in a special socket at approximately a 45- or 90-degree angle to the system board or expansion board. To install a SIMM, hold it with the chip side toward you and the gold card-edge connector pointing down and slightly away from you. Insert the SIMM's connector into the connector in the socket, *gently* pushing on the top of the SIMM until it snaps into place. You shouldn't have to apply a great deal of pressure. If you do, something may be improperly aligned.

Configuring the System

Even if you're merely upgrading the conventional memory of an 8088 or 8086, you'll probably need to reset jumpers or switches on your system board to make the machine recognize the new RAM chips. When you install expanded memory in any system (or a memory manager in a 386 or later machine), you'll need to add the memory management software to your CONFIG.SYS file, and you may need to rerun the configuration program for the expanded memory board. If you're installing your first memory expansion board, you'll probably need to set some jumpers or switches in addition to installing the RAM chips in the sockets. On a 286-class or later system, you may need to run a configuration program when you add conventional or extended memory. Before performing any of these procedures, consult your machine's documentation and that of any board you're installing.

Troubleshooting

If you bend a pin during chip installation, you'll likely receive an error message at bootup. On IBM PS/2s with Micro Channel Architecture, the system often locks out the bad area of memory and doesn't let your programs use it. The system displays an error message, but it still functions. Most other computers will display an error message and lock up if a chip malfunctions. If you're installing memory in an IBM 8088 or 80286 system, for example, the machine will flash you an error message and tell you the address of the byte with the bad chip. You might see this message:

```
201 error 0508
```

The 201 is IBM's error code for a memory failure; the 05 is a hexadecimal number that represents the first two digits of the address of the failing byte. An 8088-based IBM XT has four banks of memory: two 256K banks numbered zero and 1 and two 64K banks numbered 2 and 3. The digits 05 in the message mean that the error occurred between addresses 05000 and 05FFFF hex, or 20,480 and 24,575 decimal. Because bank 0 contains addresses zero through 262,144 (256K) decimal, the problem is in bank 0.

The 08 also is a hex number; it tells you which bit (or which chip) at that address isn't working. Each bit or chip represents a succeeding power of 2—the first chip in the bank is 00, the second is 02, the third 04, and fourth 08—so in this case, the fourth chip in bank 0 is the bad one. Knowing this, you can locate bank 0 and replace the fourth chip. (Bank 0 may be labeled as such. If not, you'll need to check your computer's documentation or use trial and error to determine which bank is which.)

Remember, IBM's 8088s and 80286s will give you this kind of information. If your compatible's manufacturer gives equally good diagnostics, you shouldn't have much trouble finding the bad chip and then correcting the problem. If your system is less efficient, a careful physical examination should reveal which pin is incorrectly seated. If you still can't pinpoint the problem, contact your machine's manufacturer.

Every computer system should have adequate memory to get the job done. With RAM prices now below $75 per megabyte, you have no reason to let lack of sufficient computer memory keep you from enjoying your system to its full potential.

David S. Veale is a computer consultant and programmer.

Take Control With Batch Files

Batch files needn't be complicated to be effective. Even one-line programs can help you take charge of your computing sessions.

BATCH FILE MAGIC

by Harry Bee

You can live your whole life without ever writing a batch file, and the sun will still shine.

On the other hand, it's easy to write useful, and even clever, batch programs, and the rewards of following this article, from the construction of simple programs to elaborate ones, are several: Batch files save you time. They save you work. And they put you in control of your computing environment.

That last bit is my favorite. I'd rather drive than ride.

One-Liners

For the most part, batch programs are lists of commands that you'd ordinarily type on the DOS command line. The commands in a batch program's list can be DOS commands or the names of programs—from major applications, such as your word processor, to the handy utilities you find in computer magazines. A batch program's list can contain just one item and still be useful.

For example, how often have you entered the directory command, DIR, watched a long directory scroll off the screen, then cursed and entered DIR /P to display the directory one screen at a time? But you can put DIR /P into a batch file called DP.BAT, and type DP instead of DIR when you want to look at a directory. Do that, and you not only type fewer keystrokes, you save yourself from another of life's little aggravations in the bargain.

The best way to learn is to do. I'll show you how to make a one-line batch program that you can use to write other batch programs. You don't need special software. Anything that produces a text file will do, and DOS comes with at least two tools that work well: the COPY command and Edlin. If you've upgraded to DOS 5.0, you also have Edit. (The sidebar, "Building Batches," describes how to use DOS's Edlin and Edit to write batch programs. It also outlines a method I use to organize my batch files.) I'll start with the COPY command.

Change to your \BATCH directory, and type the following command:

```
COPY CON CC.BAT
```

at the DOS prompt. This form of the COPY command will copy what you type on your computer console to a disk file named CC.BAT. When you press Enter, the cursor drops to the next line, which is blank. The DOS prompt has disappeared because you're no longer "in" DOS. You're in the COPY command.

On the blank line, type:

```
COPY CON %1
```

and press Enter. The cursor drops to the next blank line. In a batch file, each instruction must appear on its own line.

You're still in the COPY command, and you could continue to type more commands, one to a line. But since

CC.BAT is a one-line batch program, stop here by entering an end-of-file character by pressing the Ctrl key and the Z key at the same time, or by pressing the F6 function key. In either case, you'll see the characters ^Z on the screen. Press Enter. Your disk drive spins, the DOS prompt returns, and a batch program called CC.BAT is ready for you to use to write other short batch files.

To prove it's there, and to see what the inside of a batch file looks like, type this line at the DOS prompt:

```
TYPE CC.BAT
```

and press Enter. Because CC.BAT is a text file, DOS can type it out on the screen just the way you typed it. You can use the DOS TYPE command to look at any batch file.

Variations on a Theme

CC.BAT can create any number of different files. The device that gives CC.BAT this kind of flexibility is the variable %1. It is called a variable because, in a batch program, %1 is never itself, but refers to something else. Its value can be different every time you use the program.

Many programs and commands let you vary the way they operate by typing the command or program name and adding what is called a parameter. For example, the DOS DIR command behaves in one way, while DIR followed by the /P parameter (DIR /P) behaves in another. Similarly, most applications let you include the name of a file on the command line, so that the program starts up and loads a document or data for you to work on.

Batch programs allow the same flexibility. You can include several parameters on the command line; each one is separated from the next

BUILDING BATCHES

To create and use batch files effectively, you need a text editor with which to write them and a well-organized disk. Neither requirement is difficult to satisfy.

Get Organized

Basic hard disk organization has become a personal computing standard. If you follow the installation instructions whenever you add new software to your system, it's likely you have a separate directory for each major application you use, and a directory called \DOS holding all the programs and accessory files that come with DOS. You also should have a minimum number of files in your root directory.

If that's not the case, you have work to do. Even if it means reinstalling all your software down to the operating system, you must organize your disk. Your word processor and all the files that go with it must be in one directory branch, your spreadsheet and its files in another, your CAD program in another, and so on for all the applications you use. DOS also needs a directory of its own, and your root directory must contain CONFIG.SYS, the file DOS uses to fit itself to your computer; COMMAND.COM, the movable part of DOS; AUTOEXEC.BAT, the file DOS uses to conform itself to your liking; and a few other files.

If your root directory is cluttered, consider moving utility programs, such as screen savers, file recovery programs, and disk tools, into a \UTILITY directory. Then, add a directory called \BATCH, in which you'll keep all your batch (.BAT) files, *except* AUTOEXEC.BAT.

By bundling your DOS accessories, utility programs, and batch files into separate directories, you make them easy to locate when you need them. More importantly, you make it possible to set up your system so that you can use DOS, utility, and batch programs from any directory—by including, or modifying, the PATH command in your AUTOEXEC.BAT file. To edit your AUTOEXEC.BAT file, however, you need to choose a batch file editor.

Edlin

Since you use MS-DOS, you have Edlin. This vintage line editor has been part of the operating system's package since the beginning, and it shows its age. Its primitive, one-line-at-a-time editing has earned it a reputation as unfriendly and difficult.

Edlin's blank face definitely is unfriendly: Whatever you want from the program you must ask for. And Edlin's cryptic commands are difficult. The good news is that all you have to know about Edlin is that you refer to each line of text in a file by number, starting with 1 for the first line, and that you can write any batch file using only five of the program's commands:

- Use *i* (for *insert*) at Edlin's asterisk prompt whenever you want to add new lines to a file. Precede the i command with a line number to add lines to a file starting at that line number. Enter as many lines as you have to; the program keeps track of the numbering. When you're done, type Ctrl-C to tell Edlin to return to the asterisk prompt.
- Enter *l* (for *list*) to take a look at your file. Precede the command with the number of the line from which you want to start the listing.
- Enter a line number by itself, at the asterisk prompt, to edit that line. Here's where you can use DOS's mysterious function keys, F1 through F5, for line editing. But, the fact is, it's easier to retype the line.
- Precede *d*, for *delete*, with the number of the line you want to delete. But be sure to list the file each time, and check the current line numbers before you use this command.
- When you're finished, enter *e* (for *exit*) to leave the program and save the file to disk. (To leave the program without saving the file, enter *q* for *quit.*)

Edlin's other commands allow you to do

by a space. While a batch program runs, it uses the first parameter wherever %1 appears, replaces %2 with the second, uses the third in place of %3, and so on through %9. At the DOS prompt enter:

```
CC DP.BAT
```

On the blank line, type:

```
DIR %1 /P
```

and press Enter. Enter Ctrl-Z to return to the DOS prompt. You've just used CC.BAT to create a variation on the original DP.BAT. From now on, use DP instead of DIR to look at a directory.

Because you included the parameter variable, %1, in your batch program, you can use file masks (sequences such as *.EXE or FILE??.EXE that contain the wildcard characters * or ?) with DP, just as you do with DIR. The difference, however, is that long directories will never again scroll off the screen.

The Three-Step Formula

The most common form of batch programs is built on three instructions:

- change to a directory
- start a program in that directory
- change back to the root directory

More than half of all the batch files in creation are elaborations of that structure, and even if you've never written a batch file yourself, you probably use several of them already. They're often left behind by the process that installs a new application program on your hard disk.

Now consider all the application programs you use frequently. Are there any for which you have to change to its directory before you start it? If so, use CC.BAT to make a

much more, such as merge files, and move text within your file, but batch files are not sufficiently large or complex to warrant that level of editing. Edlin would not be the program to use if they were.

Make a PATH with Edlin

Before you modify the PATH command in your AUTOEXEC.BAT file, you may have to switch to your DOS directory by typing:

```
CD \DOS
```

to start Edlin. At the DOS prompt enter the following command:

```
EDLIN \AUTOEXEC.BAT
```

This tells DOS to start Edlin and go to the root directory to find AUTOEXEC.BAT. Type *l* at Edlin's asterisk prompt to list the file. If there's nothing in your AUTOEXEC.BAT file, the PATH command is a good place to start building one. If your AUTOEXEC.BAT file is longer than 23 lines, enter 23*l* to display more of the file, starting at line 23—and ask yourself why the file is that long!

Find the line that begins with the PATH command. If you find it, enter its line number to edit the line. If you don't find it, enter 2*i* to add the path command near the beginning of the file. Now, enter the following line (assuming your hard drive is drive C):

```
PATH=C:\;C:\DOS;C:\BATCH;C:\UTILITY
```

If you're replacing a path that contains other directories, add them to the end of the line, but find out why they're included. Your PATH statement should list as few directories as possible.

Finally, exit Edlin and save the edited file by typing *e* at the asterisk prompt. Press Ctrl-Alt-Del to restart your computer. From now on, you can use any DOS command, utility, or batch program—from any directory on your disk.

Edit

With DOS 5.0, Microsoft began including a full-screen text editor called Edit. Although Edlin is still part of the DOS package, Edit's presence diminishes the veteran line editor to the status of museum piece.

Edit is open, friendly, and easy. When you use it, the file you're working on is always in front of you. You can select all of Edit's file and editing commands from menus—with the keyboard or a mouse, if you have one. With the cursor control keys or the mouse, you can move freely through your file.

Even with the manual open, Edlin remains arcane; with Edit, the manual is hardly necessary. Where Edit isn't obviously self-explanatory, everything you need to know about it is available within the program by selecting the help menu.

Edit may not be the compelling reason to upgrade to DOS 5.0, but it's one of several significant improvements that make the new version appealing.

Beyond Dos

You can use any text editor to write batch files. You may have one that's part of a database manager or desktop publisher that you use often and already are comfortable with. Unless your word processor is a relic, you probably can use that, too. You could even use your spreadsheet program, if that's the application you use most often. The key is that the program in which you choose to write batch programs must be able to save the files, with a .BAT extension, in your \BATCH directory, as text, ASCII text, DOS text, or generic text. All of these are roughly equivalent when it comes to creating or modifying batch files.

In the end, it doesn't matter how you write the files. What's important is that you learn to build batch files, so that you can take personal control of your personal computing environment.

three-step batch program to start each of them.

For example, if you use the Word Perfect word processor, and keep it in a directory called \WP51, its start-up program looks like this:

```
CD \WP51
WP %1
CD \
```

Substitute your own word processor and directory, call the program W.BAT, and keep it in your \BATCH directory. Then, no matter what directory you're in, you can start your word processor by typing W and, if you wish, the name of a document to load.

In half an hour, using W.BAT as a model and CC.BAT to write them, you can have a three-step program to start every application you use. But don't stop there. Think about the particular ways you use your application programs. For instance, if you write a lot of memos, you ought to have a boilerplate memo document with the margins set and the headings in place. Why not also make a batch program to start your word processor to write memos? One of my variations on W.BAT, for example, looks like this:

```
CD \WP51
WP MEMO.DOC
CD \
```

Substitute your own word processor and document for mine, call the file MM.BAT, and whenever you get the urge to write a memo, simply enter MM at the nearest DOS prompt.

You can do the same thing for every computing task you perform often. If you frequently invoke your database manager to work with a particular file, call up your spreadsheet to work with a particular model, or run your communications software to connect to a particular service, each task deserves a batch program of its own. If you take the time and the small effort required to write each of these batch programs, you have then begun to make your computing environment work your way.

Getting to Bed

To do more with batch programs, however, you need a stronger tool than CC.BAT. You can use CC.BAT to make a new tool and, at the same time, get an introduction to some of the special commands DOS provides for batch programming.

Program Listing 1 lists the batch program BED.BAT (*bed* stands for *batch editor*). I chose Edlin as a text editor because, no matter what version of DOS you have, you have Edlin.

Use CC.BAT to create BED.BAT in your \BATCH directory, just as you used it to make other batch files. The only difference is that this time you'll enter a few more lines before you insert a Ctrl-Z end-of-file marker. Enter only the six lines in the listing that don't begin with a REM (for *remark*) statement. The REM lines are program notes that make the listing informative. Useful in long programs, they slow the execution of short programs noticeably.

Program Listing 1. BED.BAT is a simple batch file editor.

```
@ECHO OFF

REM            BED.BAT

REM  This program demonstrates how
REM  the Edlin line editor might be
REM  employed to edit batch programs
REM  that use the three-step formula.
REM  Substitute your own editor and pathnames.

REM  The program checks that Edlin has
REM  operated correctly, in case there's
REM  an error message you need to see.

CD \BATCH
\DOS\EDLIN %1
IF ERRORLEVEL 1 PAUSE
CD \
CLS
```

End

BED.BAT introduces some new batch programming ideas, but at its core is the three-step formula I talked about earlier. Near the bottom of the listing are the commands that change to the \BATCH directory, start Edlin with a filename you include on the command line, and return to the root directory.

One of the new ideas in BED.BAT is neatness. When you run any of the three-step programs mentioned earlier, you see that batch programs clutter your screen by displaying, or echoing, every line in the batch file as it's processed.

The first line of BED.BAT turns echoing off, and the @ character at the beginning of the line (available only since DOS 3.0) keeps the line itself from echoing. Turning echoing off doesn't affect the programs your batch files may run, however. These programs may well leave their messages on your screen. For ultimate neatness, BED.BAT's last line uses DOS's clear screen command, CLS, to leave your DOS prompt neatly on a blank screen.

The second new idea in BED.BAT is safety. If something goes wrong, you need to know about it. Most programs display an error message when something bad happens. In addition, all DOS programs, and some other programs, set a value in memory that batch programs refer to as the Errorlevel.

BED.BAT uses the IF command to be sure Edlin ran successfully. The IF command is easy to interpret. It instructs the batch program that if some condition occurs, then something else must happen. In this case, the condition is that if Edlin (invoked in the previous line) encounters a problem, it sets the Errorlevel value at 1 or more, indicating that a problem exists. If an error occurs, the batch program pauses before it clears the screen—so that you can read Edlin's error message. (Whenever DOS executes a PAUSE statement, it displays a "Press any key to continue" message and waits for your response.)

With BED.BAT in your batch toolkit, you're ready for more elaborate programs.

Tactical Maneuvers

Besides neatly starting your applications, batch programs also can take the drudgery out of computing chores. For instance, in none of its five major revisions has DOS offered an elegant way to move files from one disk or directory to another. First, you copy the files; then you delete them. Though MOVE.BAT doesn't change the process, it reduces your involvement to a single instruction (see Program Listing 2).

MOVE.BAT introduces another important batch programming idea: incorporating documentation. The fact that you create a program offers no guarantee that you'll remember how you initially wanted it to work. So it's wise to build in some instructions.

The first two IF commands in the program check to see if you entered a question mark as the first parameter to request help, or if you omitted the first parameter, which indicates you need help, since the XCOPY program must have at least a source file to work with. If MOVE.BAT finds either condition is true, it goes to the place in the program labeled :HELP.

Batch program labels are lines that begin with a colon. By skipping to the one named :HELP, MOVE.BAT bypasses the functional part of the program, the section in which files might have been copied and deleted, and instead uses the ECHO command to display instructions on how to use the batch file.

If you don't indicate you want or need help, the program attempts to copy the files, using the source and destination information you provided. If you don't supply a destination path, MOVE.BAT relocates the file or files to the current directory. Unlike the COPY command, XCOPY sets an Errorlevel value, if the command fails, and tries to tell you what went wrong. If the program detects an error in the copy function, it skips the delete function entirely and goes to the :NO_COPY label, where it displays a warning along with its instructions.

Finally, if the copy function goes well, the program goes on to delete the original files, but not before showing you what it will delete (courtesy of the DIR command) and pausing to give you a chance to change your mind and terminate the program before the files are deleted.

Think about the way you use your computer. Any time you find yourself repeating the same sequence of two or more steps, consider listing those steps in a batch file. Then, with a little practice, you can learn to add the steps that make batch programs neat, safe, and well-documented.

A Fine-Tooth Comb

COMB.BAT provides an example of another kind of program (see

Program Listing 2. MOVE.BAT safely moves files from one directory or disk to another.

```
@ECHO OFF

REM                         MOVE.BAT

REM  MOVE.BAT follows the format of the DOS COPY command, thus
REM  maintaining consistency with DOS' file-management tools. It contains
REM  a help screen and insists that you name a source file.
REM  It informs you of an unsuccessful attempt to copy the files
REM  and prompts you to agree to the deletion of the original files.

IF "%1"=="?" GOTO HELP
IF "%1"=="" GOTO HELP

XCOPY %1 %2
IF ERRORLEVEL 1 GOTO NO_COPY

DIR %1 /W
ECHO.
ECHO               WARNING
ECHO               =======
ECHO The files listed above will be deleted!
ECHO Press Ctrl-C to abort or...
PAUSE
DEL %1
ECHO.
GOTO END

:NO_COPY
ECHO.
ECHO Attempt to copy was UNSUCCESSFUL!
ECHO.

:HELP
ECHO Syntax: Move [source_file_spec] [optional_destination_path]
ECHO.
ECHO * You must supply the path and name of the source file [source_file_spec],
ECHO   which may use wildcards, name a specific file, and
ECHO   include a pathname.
ECHO.
ECHO * The optional destination path parameter may include a filename or
ECHO   wildcard mask.
:END
```

End

Program Listing 3. COMB.BAT can combine a set of files or compile selected references from a set of files.

```
@ECHO OFF

REM                COMB.BAT

REM  This program sends a series of files to the screen, a
REM  printer, a communications port, or a disk file. It can
REM  search the files and send only fragments of them. Its
REM  many uses include combining a set of files into a single
REM  archive file; search through a group of files and printing
REM  all references to a particular date, name, or other reference;
REM  or creating a document log. Although the program assumes
REM  it's working with text files, it also may work with
REM  nontext files. However, the results are difficult to
REM  predict.

REM  The program must be able to find the file CR.FIL, which
REM  contains a single, blank line. As written, the program
REM  looks for CR.FIL in the root directory.

IF "%1" == "?" GOTO SYNTAX
IF "%1" == "" GOTO MISSING
IF "%2" == "" GOTO MISSING

SET TARGET=%3
IF "%3"=="" SET TARGET=CON
DATE < \CR.FIL | FIND "is" >> %TARGET%
TIME < \CR.FIL | FIND "is" >> %TARGET%
ECHO. >> %TARGET%

IF "%2" == "$$" GOTO NO_SEARCH

ECHO Searching for %2 >> %TARGET%
ECHO.
FOR %%F IN (%1) DO FIND "%2" %%F >> %TARGET%
GOTO END

:NO_SEARCH
FOR %%F IN (%1) DO TYPE %%F >> %TARGET%
GOTO END

:MISSING
ECHO.
ECHO Parameter missing.

:SYNTAX
ECHO.
ECHO Syntax: COMB [source_file_spec] [search_string/$$] [destination]
ECHO.
ECHO * [source_file_spec] may include a wildcard mask or name filename
ECHO   and may include a pathname.
ECHO * [search_string] is a group of characters to search for in the
ECHO   source file(s). The program will display only those lines of
ECHO   the source file(s) that contain the search string.
ECHO * To prevent a search, substitute $$ for the search string. The program
ECHO   types entire file(s).
ECHO * [destination] is optional. It can be the name of a disk file;
ECHO   or the printer designation, PRN; or a COM port. If this parameter
ECHO   is omitted, the program will send the files to your monitor.
:END
```

End

Program Listing 3). Unlike my MOVE.BAT program, which gathers several steps under a single command, COMB.BAT takes one idea and lets you apply it to a whole variety of different circumstances.

The central idea of COMB.BAT is combining files. I wrote it because I periodically print a copy of all the pieces of a work in progress—from modules for a programming project to fragments of research for an article.

So COMB.BAT began as the following one-line batch file:

```
FOR %%F IN (%1) DO TYPE %%F > PRN
```

DOS's FOR command is obscure, but the line does translate: For each file (%%F) that matches the mask (%1) entered on the command line, the program sends (DO TYPE) each file (%%F) in turn to (>) the printer (PRN). Two variations of the line appear in the middle of Listing 3.

The program grew because the idea of combining files was generally useful. No sooner had I printed a set of files, than I realized I could use the same process to send a set of files to another disk file. Used on my electronic mail, once a month, it was the electronic equivalent of bundling letters with string to store in the attic. And I also could send a set of files to the screen to scan the contents of those mystery files I periodically stumble across in some dusty corner of my hard disk.

So I started to modify and expand the one-line program to accommodate my choice of destinations for the sets of files it processed. Along the way, it struck me that it also would be useful to be able to gather selected parts of a set of files.

Wouldn't it be great, I thought, to scan key-word references from a set of articles I'd collected from on-line sources, to see which files most likely had the material I needed. I discovered I could carry things further. By adding a date field with a colon to my memo heading, and searching for lines containing a colon, the program could quickly print a log of memos, before I deleted the documents from my disk.

One idea, a pocketful of uses. All it took to turn a useful one-liner of limited scope into a generally useful utility were a few embellishments on the original theme. I can't, in limited space, explain in detail all of DOS's batch file commands. That you can get from your DOS manual. But nothing about batch programming is difficult to learn, as the operation of COMB.BAT will prove as you type it.

Program Listing 4. B.BAT shows how you might create a multipurpose batch program.

```
@ECHO OFF

REM                    B.BAT

REM  An anthology of useful functions in one batch program.

REM  The program begins by checking the first parameter, %1.
REM  If the parameter is missing, or if it's a question mark,
REM  the program displays the help screen.

IF "%1"=="" GOTO HELP_SCREEN
IF "%1"=="?" GOTO HELP_SCREEN

REM  Otherwise, it branches to the function called for by %1.
REM  (If %1 is invalid, the program quits and displays the
REM  error message, "Label not found.")

GOTO %1

REM  Below is a simple paged directory function. You get
REM  here by entering
REM  B D [optional path- and filename]
REM  on the command line.
REM  Once the program finds this place, by substituting
REM  D for %1 in the GOTO instruction, above, the SHIFT command
REM  moves all the command-line parameters left so that the
REM  DIR command, below, then uses the optional path- and
REM  filename, if you included it, for %1. All the functions
REM  in the program use this parameter shift.

:D
SHIFT
DIR %1 /P
GOTO END

REM  This function invokes COPY CON for creating simple text
REM  files. Since the COPY command will overwrite an existing
REM  file, the program checks the current directory for duplicate
REM  filenames and lets you decide whether to quit or continue.

:C
SHIFT
IF "%1"=="" GOTO HELP_SCREEN
IF NOT EXIST %1 GOTO COPCON
ECHO.
ECHO %1 EXISTS. PRESS CTRL-C TO QUIT OR
PAUSE
:COPCON
ECHO.
ECHO Editing %
ECHO Press Enter at the end of each line, then press ^Z or F6 to end.
ECHO - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
COPY CON %1
GOTO ROOT

REM  This substitute for the dangerous DOS DEL command shows you
REM  the files you're about to delete and gives you a chance to
REM  change your mind. As a bonus, it repeats itself until there
REM  are no more filenames or masks left. Thus, if you enter:
REM               B X *.BAK *.TMP *.DAT
REM  the function will delete (from the current directory) all
REM  files with a .BAK extension, then all those with a .TMP
REM  extension, then all with a .DAT extension.

:X
SHIFT
IF "%1"=="" GOTO END
DIR %1 /W
ECHO.
ECHO You are about to delete these files!
ECHO Press Ctrl-C to keep them, or...
PAUSE
DEL %1
GOTO X

REM  This function prevents you from accidentally reformatting
your
REM  system's drive C. Keep FORMAT.EXE and other dangerous pro-
grams REM  in a subdirectory that isn't named in your system's
PATH
REM  command. Below I used \SAFE. If you don't include the name
REM  of a drive to format, the function defaults to drive A,
REM  rather than to drive C, which most people use as the current
REM  drive.

:F
SHIFT
SET DR=%1
IF "%DR%"=="" SET DR=A:
\SAFE\FORMAT %DR%
SET DR=
GOTO ROOT

REM  This is the help screen. It shows all the valid function
REM  labels and any other parameters needed or permitted.

:HELP_SCREEN
CLS
ECHO Syntax: b [function_label] [function_parameters...]
ECHO.
ECHO Valid Function Labels
ECHO - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
ECHO D [optional_\path\filespec]     Paged directory
ECHO C [program.ext]                 COPY CON file creator
ECHO X [filespec_masks...]           Safe file deletion
ECHO F [optional_drive:]             No-calamity disk formatting
ECHO.
GOTO END

REM  End functions here to finish in the root directory.

:ROOT
CD \

REM  End here to stay in the current directory.

:END
```

End

The first line turns off echoing. (Skip the remarks.) The next three IF commands check to see if you want help, or obviously need it. The next line sets an "environment" variable (TARGET) equal to the destination device, or filename supplied as the third parameter on the command line. The next sets the environment variable to CON (your monitor), if you didn't supply a destination.

The next two lines put the date and time on the document or file you're making. So that you don't have to sit at your computer to press the Enter key, which the commands need to finish, both use a carriage return from (<) a file you put on your disk (more on that in a moment). The program filters (|) the output of the DATE and TIME commands through DOS's FIND program, so that only the lines containing the date and time go to (>>) your target destination. The next line sends a blank line to your target drive and directory.

The subsequent line checks the

search term you supplied as the second command line parameter to see if it is the unique signal ($$) that indicates you want to process entire files. If it finds $$, COMB.BAT sends you ahead to the label :NO_SEARCH.

The next two sections essentially run two variations of the original program idea. The first uses DOS's FIND program to pick out fragments of files according to a search term you supply; the second sends entire files to your target device or file. Both versions skip to the end of the program when they're done.

Now, about that file containing a carriage return: COMB.BAT depends on the file CR.FIL, which you can easily create with CC.BAT. At the DOS prompt, type:

```
CC \CR.FIL
```

On the first unprompted line, press Enter. On the next unprompted line, terminate the file with Ctrl-Z. This creates a two-byte file containing a single carriage return.

It's Your Turn

Batch programs don't have to be long, or even pretty, to be effective. As in every skill you learn, grace comes with practice. The first step toward taking control of your computing is to get into the habit of noticing the things you do at the DOS prompt, especially the tasks you repeat. Then take the minute necessary to write a habitual chore into a short batch file. Later, you can come back to improve the program.

Writing batch programs is easy, but challenging enough to be fun. Start writing them, and you soon find yourself with a directory full of files getting difficult to manage. Program Listing 4, B.BAT, offers a way to collect all your short batch files, your convenient one-liners and three-step application launchers, into one master program. I've included two of the one-liners discussed above and added two more handy utilities. As you add your own functions to the program, be sure to document each one in the help section at the end.

Now, get behind the wheel. It's your turn to drive.

Free-lance writer Harry Bee has been working with computers for 12 years.

Tip 6

Delete with Impunity

If you've ever accidentally erased important files with a hastily entered DEL command, you'll appreciate the following fail-safe file-deletion technique. Start by using the DIR command to list the files you think you want to erase. If the resulting directory listing contains only files you want to get rid of, type DEL, press the F3 function key, then press Enter.

This trick works because DIR and DEL have the same number of letters and because, when you press F3, DOS recalls only the part of the previous command line that is to the right of the current cursor location.

Tip 7

Eliminating Disk Clutter

To avoid clutter, the root directory of your hard disk should contain the fewest number of files possible. Certainly, the hidden DOS system files and COMMAND.COM belong there, as do your CONFIG.SYS and AUTOEXEC.BAT files. But what about device drivers invoked in CONFIG.SYS?

Suppose, for example, that you use the DOS ANSI.SYS device driver to change the default colors on your display. If you place the simple command DEVICE=ANSI.SYS in your CONFIG.SYS file, then the ANSI.SYS driver would have to be stored in the root directory of your hard disk. If you do the same for your mouse driver and other drivers, you'll soon end up with a cluttered root directory. To avoid the problem, try using the DOS MD command to set up a special directory, DRIVERS, to hold all the device drivers you use during system startup.

Setting up a special device driver directory in this manner has two advantages. First, it cleans up your hard disk's root directory. It also organizes all your device drivers in a single directory, making them easier to keep track of.

Because DOS allows you to include pathnames in your CONFIG.SYS file, including the following line in your CONFIG.SYS file would allow DOS to find and load ANSI.SYS from your special device-driver directory, each time you start your system:

```
DEVICE=C:\DEVICE\ANSI.SYS
```

Elementary Steps To Programming

If you have DOS 5.0, you have QBasic. Now's the time to learn how to use it for writing programs.

PROGRAMMING

by Hardin Brothers

The first step in any project is to design it. You also need to be able to use the tools for your job; to communicate ideas to others effectively; and to organize and carry out your plan, sometimes working on separate parts before combining them into the finished project.

So it is with programming. Your first step is to sit down with paper and pencil and design a program (the project). You must be able to use a text editor, interpreter or compiler, and debugger (the tools); communicate your information logically (use the language); and organize and carry out your plan (write the program), usually working on several levels at a time.

If you keep these three concepts in mind—tools, language, and writing what you've designed—and learn to separate them from one another, you will find that programming is not as overwhelming a task as you may have thought. In this article, I walk programming beginners through the development of a short phone list utility program to provide a feeling for how a language and its idioms and syntax work. The program will display names and phone numbers on the screen one page at a time, and it will let you move up or down through the list and add names and numbers or delete them from the list. And it will divide your screen into two parts: The top and largest part will display a sorted list, and the bottom will display a menu of allowable keystroke commands and space for new entries.

The Programming Tools

A software program that creates, edits, and manages text files—the actual words of the language you use for writing a program—is a text editor. A software program that translates the text of a programming language into machine language, code that the computer understands, is an interpreter or compiler. And a debugger is a program that helps to correct errors in your program logic.

When you are learning to program, you should choose tools that will help you. Pick a programming environment that has all the above-mentioned programs built in. Our phone list utility uses the QBasic interpreter that comes with MS-DOS 5.0. You also can use Microsoft Quick Basic 4.0 or 4.5. The program we're about to write works fine with Quick Basic, but this article will deal with QBasic. Though far from perfect, QBasic makes programming nearly painless. And even if it's not the best choice for large projects, it is superb for getting small programs running quickly. QBasic is a far more capable language than GW-Basic or BasicA, which were included with earlier versions of DOS.

Speaking of Language

Why QBasic? First of all, it's included free with DOS 5.0. When you upgrade to DOS 5.0—and you should—QBasic comes along as a bonus. That's just one more reason why you should be running the latest version of DOS. Also, as a close relative of Microsoft's Quick Basic, QBasic is one of the easiest general-purpose languages to learn and to use. The most obvious difference between Quick Basic and QBasic is that Quick Basic is a compiler, which means you can use it to create stand-alone .EXE programs. QBasic, on the other hand, is an interpreter, and that means the programs you write can run only from within the QBasic environment.

QBasic allows and encourages you to use modern structured programming techniques. Structured programming techniques can help you deal with the complexities of a program because they apply a logical structure to your program writing. Specialized CASE (computer-assisted software engineering) and structured analysis tools also provide help, but these products are too expensive and too complex for programs of fewer than 5,000 lines. For small projects, your own style of working—with notes, pseudocode, or flow charts—should keep you from getting lost while you are concentrating on the details of a program.

No matter which language you use, you have to learn its syntax and idioms. Syntax is the set of rules that governs the form of your program statements and the relationship among your words. Idioms, the usual way in which the words of a particular language are joined to express a thought, are more difficult. You can learn the idioms of a language by reading code written by good programmers until you understand why each statement was made in preference to possible alternatives.

You're Ready To Write a Program

With a design in mind, it's time for you to fire up the QBasic interpreter by typing:

```
QBASIC
```

at the DOS prompt. Set the display to your liking first: Pull down the Options menu, select Display, and set the tab stops to something reasonable, such as 3. Set the display colors, if you wish, and select OK from the bottom of the screen. QBasic will save your options in a file called QBASIC.INI. Each time you run QBasic, it will read the QBASIC.INI file and start up with the same options in effect.

Now you can begin writing the top-level module of the PHONE.BAS program—the code that controls how the rest of the program operates (see Program Listing 1). The first line of almost every QBasic program is DEFINT A–Z. This line tells QBasic that you want to use integers as a default setting. Unless you tell QBasic this, it assumes that all numeric variables should be represented internally in floating-point (decimal) form. Manipulating floating-point numbers is relatively slow, especially if you don't have a math coprocessor chip in your computer. The program will run much faster if most of its numeric variables are stored as integers.

Most of the top-level code is a series of definitions. The program defines a new data type called EntryType, which will be used to hold information on each person in the phone book. The series of CONST declarations defines fixed values at the top of the program, where you can make changes if you don't like, for example, the way the screen is divided between the phone book display and the menu or the number of allowable entries in the phone book.

The lines that begin with a DIM SHARED statement define the only set of variables that will be available throughout the entire program. All other variables will be visible only within the procedure that uses them, so there won't be unexpected side effects from a conflict of variable names.

The general operation of the program is contained in three lines ReadFile (which opens and reads in the data file), MainDisplay (which displays the phone book and menu), and SaveFile (which saves the phone book data for the next time you use the program). The last three lines of the top-level code compose a generic error trap, which simply puts an error number in a global variable (one used by all program modules) and then returns to the routine that caused the error.

To write this part of the program, simply type in the lines one by one. Once you have typed in all of Listing 1, pull down the File menu and select Save. You will be asked to enter a filename. Type in PHONE.BAS and press Enter.

The Support Routines

If you wrote this program in true top-down style, you would next create the code for the three procedures that are called from the module level: ReadFile, MainDisplay, and SaveFile. Each of these procedures might call others, which you would then write, and these sub-subprocedures might call others. When every call had a related chunk of code to call, you would be finished.

Some programmers prefer to work from the bottom up. They start by writing the lowest-level support routines and gradually work their way up to the module-level code.

Many programmers work from both the top and bottom, creating some support routines and then writing the higher-level routines that call them. Working like this requires that you keep more of the program in your head or on paper, but it seems to be a more natural way of handling small projects. In any case, this is how we will handle the phone book utility project.

Program Listing 2 contains a handful of functions and subprograms that provide low-level support for the rest of the program. The first, YesNo$, prompts you to type a *y* or *n*, waits until you comply with the prompt, and then returns the uppercase version of your response to the calling routine. You can use YesNo$ in a variety of programs.

FileExists is another general-purpose function. It receives the name of a file and tests whether the file exists. It returns a true or false value to the procedure that called it. A more sophisticated version of FileExists would use DOS calls to avoid relying on error trapping, but otherwise, it would look the same as the one in Listing 2. As with YesNo$, you could use FileExists$ in other programs.

GetKey is a third general-purpose routine. It waits for you to press a key and returns the key's ASCII value to the program. If you press a function key or other special key, the Basic function Inkey$ will return a two-character string with the first character equal to ASCII 0. To make it easier for the program to handle those keys, GetKey multiplies the ASCII value of the second character by –1 and returns a negative value.

Finally, ClearTop and ClearBottom are subprograms that simply clear the top part and bottom part of the screen, respectively.

Program Listing 1. This top-level module of the PHONE.BAS program contains the code that controls the operation of the rest of the program. Combine it with Listings 2 and 3 to make up the complete program.

```
DEFINT A-Z

TYPE EntryType
   LastName AS STRING * 20
   FirstName AS STRING * 20
   Phone AS STRING * 14
END TYPE

CONST RecLen = 54             'Length of EntryType
CONST FALSE = 0
CONST TRUE = NOT FALSE
CONST FormLength = 18         'Lines in screen top
CONST MaxEntries = 500        'Maximum number of entries
CONST FileName$ = "PHONE.LST" 'Data filename

CONST PgUp = -&H49
CONST PgDn = -&H51
CONST Esc = 27

DIM SHARED Entry(1 TO MaxEntries) AS EntryType
DIM SHARED CurrentEntry, LastEntry
DIM SHARED ErrorFlag
DIM SHARED BlankLine$, DashLine$

BlankLine$ = STRING$(79, " ")
DashLine$ = STRING$(79, "=")

ReadFile
MainDisplay
SaveFile
CLS
END

ErrorTrap:
   ErrorFlag = ERR
   RESUME NEXT
```

End

QBasic makes writing the functions and subprograms very easy. To create a new function, such as YesNo$, pull down the Edit menu and select New Function. QBasic will ask you to name the new function, and you should type in YesNo$ and press Enter. QBasic will then display the following lines:

```
DEFINT A-Z
FUNCTION YesNo$
END FUNCTION
```

Your job is to type in the lines that go between FUNCTION YesNo$ and END FUNCTION. After doing so, pull down the File menu and select Save to preserve your work. Go on to the next function by pulling down Edit, selecting New Function, and typing in FileExists (File$). Fill in the lines between the start and end of the function and save your work as before.

Following the same steps, type in each of the functions and subprograms. When writing subprograms, select New Sub rather than New Function from the Edit menu. Subprograms begin with the SUB statement, while functions begin with FUNCTION.

It is very easy to make mistakes when typing in program code, so check your work carefully. The spelling of each item is important, and so are punctuation marks such as commas, colons, and semicolons. A single mistyped variable name or one missing punctuation mark can bring the program to a crashing halt when you try to run it. Capitalization is generally not important, except to make the listings easier for you to read.

The Heart of the Program

The module-level code and low-level support routines are done. It's time to work on the heart of the program: the three subprograms that are called by the module-level code.

Program Listing 3 begins with the first two of these subprograms, ReadFile and SaveFile. They should be designed so that data is stored and retrieved in the same way. The third subprogram, MainDisplay, begins the real work of the program. It clears the screen and turns off the cursor. Next it calls ShowMenu to display the menu at the bottom of the screen

Program Listing 2. This is the second part of PHONE.BAS, which provides low-level support routines (functions and subprograms) that make sure everything is set up for the program to run properly.

```
FUNCTION YesNo$
   PRINT " (y/n) ==> ";
   DO
      Ch$ = UCASE$(INPUT$(1))
   LOOP UNTIL Ch$ = "Y" OR Ch$ = "N"
   PRINT Ch$
   YesNo$ = Ch$
END FUNCTION


FUNCTION FileExists (File$)
  ErrorFlag = 0
  FileNum = FREEFILE
  ON ERROR GOTO ErrorTrap
  OPEN File$ FOR INPUT AS FileNum
  ON ERROR GOTO 0
  CLOSE FileNum
  IF ErrorFlag = 0 THEN
     FileExists = TRUE
  ELSE
     FileExists = FALSE
  END IF
END FUNCTION


FUNCTION GetKey
  DO
     Ch$ = INKEY$
  LOOP UNTIL LEN(Ch$) > 0
  IF LEN(Ch$) = 1 THEN
     GetKey = ASC(UCASE$(Ch$))
  ELSE
     GetKey = -1 * ASC(RIGHT$(Ch$, 1))
  END IF
END FUNCTION


SUB ClearTop
   LOCATE 1, 1
   FOR Lp = 1 TO FormLength
      PRINT BlankLine$
   NEXT Lp
END SUB


SUB ClearBottom
   LOCATE FormLength + 1, 1
   FOR Lp = FormLength + 1 TO 23
      PRINT BlankLine$
   NEXT Lp
   PRINT BlankLine$;
END SUB
```

End

and enters a never-ending loop, called ShowList, to display a page of the phone book. Then it calls GetKey and waits for you to press a key.

MainDisplay uses a SELECT CASE statement to read your key press and decide what to do next. If that key press is PgUp or PgDn, the variable CurrentEntry is updated so that ShowList shows the correct page of the phone book when the loop repeats. If you press *A* or *D*, the program calls AddEntry or DeleteEntry, as appropriate. If you press Esc, the subprogram MainDisplay ends so that the data can be saved and the program can terminate.

MainDisplay calls four new subprograms: ShowMenu, ShowList, AddEntry, and DeleteEntry. The first two are simple procedures that clear the screen and print. AddEntry and DeleteEntry are slightly longer because each has a small amount of error-checking to do, but neither is complicated.

When a new name is added to the list, it must be placed in the right position. The final routine, SortList, does the sorting by using one of the most simple sorting algorithms possible. However, it also is relatively inefficient, and you might want to substitute a better sorting algorithm if your phone list grows to more than 100 entries.

Create each of the subprograms shown in Program Listing 3, following the same steps we used for the functions and subprograms in Listing 2. Select New Sub from the Edit menu, enter the name of the subprogram, then type in and save the lines of code that make up the subprogram.

If you forget which functions or subprograms you have typed in, select SUBS from the View menu. QBasic will display a list of all the functions and subprograms you have already entered. From this display, you can select any of the functions or subroutines for additional editing.

Putting It All Together

Although the program is shown as three listings for the purposes of explanation, it is really a single program inside the QBasic environment. As you type the functions and subprograms, QBasic will generate the correct DECLARE statements for the program and place them at the beginning of the module-level code. As you can see, QBasic performs a great many programming chores for you automatically.

Once you type the program into QBasic, you probably will have to do some debugging. Always save your program before you start to debug it; if something goes wrong and the

Program Listing 3. The heart of PHONE.BAS, this final section consists of the three subprograms that manage the details of running and using the program.

```
SUB ReadFile
   IF FileExists(FileName$) THEN
      OPEN FileName$ FOR RANDOM AS #1 LEN = RecLen
      LastEntry = LOF(1) \ RecLen
      FOR Lp = 1 TO LastEntry
         GET #1, , Entry(Lp)
      NEXT Lp
      CLOSE #1
      IF LastEntry > 0 THEN
         CurrentEntry = 1
      ELSE
         CurrentEntry = 0
      END IF
   ELSE
      CurrentEntry = 0
      LastEntry = 0
   END IF
END SUB


SUB SaveFile
   IF LastEntry > 0 THEN
      OPEN FileName$ FOR RANDOM AS #1 LEN = RecLen
      FOR Lp = 1 TO LastEntry
         PUT #1, , Entry(Lp)
      NEXT Lp
      CLOSE #1
   END IF
END SUB


SUB MainDisplay
   CLS
   LOCATE , , 0          'Turn off cursor
   ShowMenu
   DO
      ShowList (CurrentEntry)
      SELECT CASE GetKey
         CASE PgUp
            CurrentEntry = CurrentEntry - FormLength
            IF CurrentEntry < 1 THEN
               CurrentEntry = 1
            END IF
         CASE PgDn
            IF CurrentEntry + FormLength <= LastEntry THEN
               CurrentEntry = CurrentEntry + FormLength
            END IF
         CASE ASC("A")
            AddEntry
         CASE ASC("D")
            DeleteEntry
         CASE Esc
            EXIT SUB
         CASE ELSE
      END SELECT
   LOOP
END SUB


SUB ShowMenu
   ClearBottom
   LOCATE FormLength + 1, 1
   PRINT DashLine$
   PRINT "   Commands:    <PgUp>         <PgDn>  "
   PRINT "                <A>dd Name     <D>elete Name"
   PRINT "                <Esc> to end"
   PRINT DashLine$
END SUB


SUB ShowList (Start)
   ClearTop
   LOCATE 1, 1
   IF Start > 0 THEN
      FOR Lp = Start TO Start + FormLength - 1
         IF Lp > LastEntry THEN EXIT FOR
         PRINT USING "###_. "; Lp;
         PRINT Entry(Lp).LastName$; " ";
         PRINT Entry(Lp).FirstName$; " ";
         PRINT Entry(Lp).Phone$
      NEXT Lp
   END IF
END SUB


SUB AddEntry
   ClearBottom
   LOCATE FormLength + 1
   PRINT DashLine$
   IF LastEntry < MaxEntries THEN
      PRINT "Last Name ==> ";
      LINE INPUT Name$
      IF LEN(Name$) < 1 THEN
         ShowMenu
         EXIT SUB
      ELSE
         LastEntry = LastEntry + 1
         Entry(LastEntry).LastName$ = Name$
         PRINT "First Name==> ";
         LINE INPUT Entry(LastEntry).FirstName$
         PRINT "Phone     ==> ";
         LINE INPUT Entry(LastEntry).Phone$
         SortList
      END IF
   ELSE
      PRINT "The phone list is full"
      PRINT "Press any key to continue";
      Z$ = INPUT$(1)
   END IF
   IF CurrentEntry = 0 THEN CurrentEntry = 1
   ShowMenu
END SUB


SUB DeleteEntry
   ClearBottom
   LOCATE FormLength + 1
   PRINT DashLine$
   PRINT "Entry number to delete ==> ";
   LINE INPUT Item$
   Item = VAL(Item$)
   IF Item > 0 AND Item <= LastEntry THEN
      PRINT Entry(Item).LastName$; " ";
      PRINT Entry(Item).FirstName$; " ";
      PRINT Entry(Item).Phone$
      PRINT "Delete this entry";
      IF YesNo = "Y" THEN
         FOR Lp = Item TO LastEntry - 1
            Entry(Lp) = Entry(Lp + 1)
         NEXT Lp
         LastEntry = LastEntry - 1
      END IF
   END IF
   ShowMenu
END SUB
```

Listing 3 continued

Listing 3 continued

```
SUB SortList
   DIM SortChart$(0 TO 3)
   SortChart$(0) = "\": SortChart$(1) = CHR$(179)
   SortChart$(2) = "/": SortChart$(3) = CHR$(196)
   ClearBottom
   LOCATE FormLength + 1, 1
   PRINT DashLine$
   PRINT "    Sorting    ";
   IF LastEntry >= 2 THEN
      FOR Lp1 = 1 TO LastEntry - 1
         FOR Lp2 = Lp1 TO LastEntry
            IF Entry(Lp1).LastName$ > Entry(Lp2).LastName$ THEN
               SWAP Entry(Lp1), Entry(Lp2)
            END IF
         NEXT Lp2
         LOCATE CSRLIN, POS(0) - 1
         PRINT SortChart$(Lp1 MOD 4);
      NEXT Lp1
   END IF
   ShowMenu
END SUB
```

End

computer hangs up, you can reboot and reload the program, instead of typing in all the code again.

Run the program by pulling down the Run menu and selecting Start. If there are syntax errors in the program listing, QBasic will halt the program and display a portion of the listing with the error highlighted. Consult the listings printed here to find your mistake, then make the correction, save your work, and try running the program again.

More challenging to find are logic errors. While a syntax error will halt the program, a logic error will simply produce undesired results. As an example, look at the subprogram ShowList in Program Listing 3. The PRINT lines in this subprogram display the names and phone numbers on your screen. The first three PRINT lines end with semicolons. Omitting any one of these semicolons will result in the names and numbers being displayed incorrectly. As far as QBasic is concerned, no syntax errors will be detected, but the display will not be the one you want. Finding logic errors becomes easier with experience, but in the beginning, the best course is to carefully compare the listings in this magazine to what you've typed.

With a short program such as this one, you can trace through the entire listing line by line in QBasic and make sure that everything works. For longer programs, you probably won't want to spend the time to trace the entire program. Instead, set break points at the beginning of each routine you suspect might have bugs, let the program run to each break point, and then trace from there, examining variables as you go.

The best way to understand any program is to get it running and then make changes. This program allows room for many enhancements—including, for example, a better sorting routine, more attractive screen displays, a search command to find a specific name or display page, longer data records to include addresses, a print function to send the phone list to your printer, or a dial function to let your modem call a number for you. You undoubtedly can think of other changes that you would like to make to the program.

You can effect all of the changes I mention simply by altering one or two procedures in the program, or writing a new procedure and adding another CASE statement to MainDisplay and another command to ShowMenu. Experiment with the program and have fun with it. You'll learn a lot about QBasic and programming in general. ■

Free-lance writer Hardin Brothers has been working with and writing about computers for 13 years.

Tip 8

DOS Dress Up

Regardless of what type of display you have, DOS 5.0's setup program initially configures the shell to run in text mode. In text mode, for example, brackets ([ ]), rather than graphical icons, are used to indicate disk drives. To switch to a display that lets you take full advantage of your system's graphics capabilities: press Alt-O to choose Options from the DOS Shell menu and press D to modify the display.

This calls up the dialog box for the shell's screen-display mode. The specific options listed in this dialog box vary, depending on what type of monitor you have. To change the DOS shell display to true graphics mode, use the arrow keys to highlight the graphics display mode you want, then press Enter. The DOSSHELL program redraws your display using the specified graphics mode, which is now the default setting.

Seven Programs To Safeguard Your Files

Worried about file security? Use these programs to protect your precious data from loss or theft.

by Larry Sessions

The concept of PC security is almost a contradiction in terms. PCs simply are not designed with security in mind. Your files, whether stored on floppies or on a hard disk, are an open book that anyone with enough knowledge to turn on the computer and boot MS-DOS can peruse, destroy, or even steal.

Although thievery constitutes one threat to your data, it doesn't take someone with malicious intent to destroy files. Friends or associates might inadvertently erase a file, alter text, or even reformat your hard disk.

Fortunately, there are precautions you can take. In this article, I'll outline seven programs you can use to guard against accidental data loss or theft. The programs work, except as noted, with MS-DOS 2.0 and later.

A word of caution: Anyone well-versed in DOS, or with access to The Norton Utilities, PC Tools, or a similar utility, can access your data no matter how many safeguards you install. The only way to ensure that your data is completely secure is to keep original files (and their backups) on floppies and store them in a safe place.

Preventing Reformats

You've undoubtedly heard the horror stories about users who have accidentally reformatted their hard drives and lost all of their data. Although DOS 3.0 and later issue a warning message before initiating a format, hard disk users still manage to destroy data in this manner.

The easiest way to prevent accidental reformatting is to rename FORMAT.COM so that neither you, nor anyone else, is likely to invoke it unintentionally. To be safe, you should do this on your hard disk directory where the file resides and on any DOS disk that contains the file. Name it whatever you want (using the DOS RENAME command), as long as the new name is one you can remember and one that an intruder is not likely to identify. To rename it XXX.COM, for example, you'd type:

```
RENAME FORMAT.COM XXX.COM
```

It is a good idea to move this file from the root or DOS directory on your hard disk into a utility subdirectory. You can even hide it so that it doesn't appear on any directory.

After renaming FORMAT.COM, create a batch file called FORMAT.BAT (see Program Listing 1). FORMAT.BAT performs the format function and gives you, or anyone using your machine, plenty of chances to escape before damage is done. The batch file in Listing 1 assumes that the FORMAT.BAT and XXX.COM files are in the same directory on drive C and that the directory is listed in your PATH statement. You need to change the drive designations if you work on a floppy-based system. The %1 parameter represents the drive you want to format.

Once you install FORMAT.BAT, no one can accidentally reformat your hard drive or even a floppy disk. If you or someone else tries to format your hard drive, a message appears saying the procedure has been aborted. If you designate your hard drive as something other than C, or if you have more than one, you can make the appropriate adjustments to the FORMAT.BAT file.

To format drive A, go to the drive and directory that holds your FORMAT.BAT file and enter:

```
FORMAT A
```

Note that you do not enter a colon after the drive designator.

Preventing Erasure

Protecting your files from accidental erasure is as important as avoiding an accidental reformat. If you use MS-DOS 3.0 and later, one way to prevent accidental erasures is to change the file attribute to "read only."

Make sure you are in the same directory as ATTRIB.EXE. Then enter:

ATTRIB + R *filename*

for whatever file you want to protect. The file will then be protected against erasure or change. If you want to make changes to the file, enter:

ATTRIB – R *filename*

This type of protection can be a bother if you still want to make changes to the file. To solve the problem, you can rename the COPY, DELETE, and ERASE commands so that only you know how to modify or remove a file. For instance, you might change the commands to REDO, CUT, and CANCL. then if someone tries to perform COPY, DEL, or ERASE functions on one of your files, the message "Bad command or filename" will appear.

Editor's Note: Be extremely careful if you attempt to modify COMMAND.COM as described below! Never modify your original copy of COMMAND.COM. Do not modify the copy of COMMAND.COM on your hard drive until you are absolutely certain that the floppy-based version is working perfectly as described below. If in doubt, do not attempt this modification.

You cannot simply rename these commands, however. They are not files, but rather internal commands found in your COMMAND.COM file. If you have a file utility such as The Norton Utilities, you need only find the words COPY, DEL, and ERASE in COMMAND.COM and replace them with REDO, CUT, and CANCL, respectively. The command names can be replaced with any word using all uppercase letters and the same number of characters as the original. Be careful not to rename a file using a DOS reserved word or one you might use as a filename at a later time.

If you don't have one of these utilities, a powerful utility called DEBUG.COM, which is included with DOS, can do the job. It's a bit harder to use DEBUG, but if you proceed with caution, you should have no trouble. I'll run through the steps with the COMMAND.COM that comes with DOS 3.1; the procedure is essentially the same for other DOS versions.

Changing Command Names

```
1     A:\>DEBUG COMMAND.COM      {NOTE: ENTER AND WAIT FOR "-" PROMPT}
2     -S CS:100 FFFF 'COPY'
3     XXXX:4D48
4     XXXX:8488
5     -E4D48 52 45 44 4F
6     -E8488 52 45 44 4F
7     -S CS:100 FFFF 'DEL'
8     XXXX:4D32
9     XXXX:8472
10    -E4D32 43 55 54
11    -E8472 43 55 54
12    -S CS:100 FFFF 'ERASE'
13    XXXX:4D29
14    XXXX:8469
15    -E4D29 43 41 4E 43 4C
16    -E8469 43 41 4E 43 4C
17    -W     {ENTER THIS AND WAIT UNTIL THE DRIVE STOPS}
18    -Q
```

First, format a blank disk with the FORMAT /S command (or XXX /S if you have previously renamed FORMAT.COM). This formats the disk and makes it self-booting. Then copy DEBUG.COM onto the newly formatted disk. The directory of the disk should now contain COMMAND.COM and DEBUG.COM. Make sure you are in the drive that contains the new disk—*not* your hard drive if you have one—and enter the data as shown in the figure "Changing Command Names" above. Do not enter the line numbers or the comments in the curly braces ({ }).

In lines 2, 5, 6, 7, 10, 11, 12, and 15 through 18, DEBUG will give you the dash prompt. Type in the information and press Enter. Lines 3, 4, 8, 9, 13, and 14 are supplied by DEBUG. Don't worry about the "XXXX" notations. They appear as hexadecimal (hex) numbers on screen and vary among different systems. The hex numbers following them are different in other DOS versions. It's these numbers—the ones on the right side of the colons in lines 3, 4, 8, 9, 13, and 14—that you need to notice if you are using something other than DOS 3.1. For example, let's say you're using another version of DOS, and lines 3 and 4 appear on your screen like this:

```
XXXX:3C42
XXXX:8408
```

For lines 5 and 6, you would enter:

```
-E3C42 52 45 44 4F
-E8408 52 45 44 4F
```

You would follow the same procedure for lines 10, 11, 15, and 16. (By the way, the *E* in line 5 stands for *Edit*, 3C42 is the start of a memory location where the word *COPY* is stored, and 52 45 44 4F is the hex equivalent of *REDO*, your new word for the COPY command. The *W* writes the new data to COMMAND.COM, and *Q* quits DEBUG and returns you to DOS.)

Next, reboot your computer with this disk and make sure the new commands works. If not, reboot with another DOS disk or your hard drive and repeat the procedure. Once it works, store an unaltered copy of COMMAND.COM on a disk and then copy your new COMMAND.COM onto your hard disk and your self-booting floppies. If you have trouble with applications, such as word processors or spreadsheets, you may need to change back to your original COMMAND.COM. I haven't found any conflicts with my software, but it is possible.

The program should protect your files from almost anyone who tries to alter or delete them. It should also prevent someone from copying your copyrighted software without your knowledge. Although getting around these changes is fairly simple, the way to do so probably wouldn't occur to the casual user. An experienced user would know that your machine could be rebooted from an unaltered DOS disk. Therefore, you should alter any and all bootable disks. If you don't want to alter a particular disk, then put it in a locked place.

Preventing System Access

If you own DOS 3.0 or later, a simple procedure can add a low level of security. It not only will prevent the casual user from accidentally erasing or changing your files, but also will keep out prying eyes. Again, a knowledgeable user can easily discern the procedure, but it should discourage casual snooping. Make the following changes to a bootable floppy disk, test it, and when you're sure it works, transfer the changes to your hard disk and bootable floppies.

Program Listing 1. FORMAT.BAT is a batch file that controls the format function of your renamed FORMAT.COM file.

```
@ECHO OFF
CLS
ECHO    WARNING  -  Formatting a drive will destroy all data!
ECHO.
ECHO.
IF "%1" == "" GOTO END
IF "%1" == "C" GOTO END
IF "%1" == "c" GOTO END
%1:
CD\
DIR/W
ECHO.
ECHO    Are you sure you want to format this drive?  If so, press
ECHO    any key.  To abort, press Ctl-C now.
ECHO.
PAUSE
C:
CD\
REM (Note 1 - This returns you to the C drive root directory.
REM Replace "C" with "A" if you use floppies or with whatever
REM path is appropriate to find the batch file again.)
xxx %1:
REM (Note 2 - This assumes that you have renamed FORMAT.COM as
REM XXX.COM and that it resides in the same directory as this
REM FORMAT.BAT file.  Add the appropriate path if it is on another
REM drive or directory.  "%1" indicates the drive you wish to format.)
GOTO END2
:END
ECHO.
ECHO    Sorry - I'm not allowed to format drive %1:.  Procedure aborted.
ECHO.
:END2
```

End

Program Listing 2. The DEBUG script file, PASS.SCR, creates a passkey-protection program for securing data.

```
A 100
MOV     BH,26    ; sets "&" as passkey
PUSH    AX       ; clear screen routine
MOV     AX,2
INT     10
POP     AX
MOV     AH,9     ; print message routine
MOV     DX,011A
INT     21
MOV     AH,7     ; get keyboard input, store in AL
INT     21
SUB     AL,BH    ; see if keyboard input is &
JNZ     0110     ; loop if keyboard input is not &
INT     20       ; return to DOS
DB 'Please enter passkey:$ '

RCX
30
N PASS.COM
W
Q
```

End

Program Listing 3. The DEBUG script file, HALT.SCR, creates a passkey-protection program that lets you supply a passkey of up to 118 characters.

```
A 100
MOV     AH,25   ; set interrupt for Ctl-C/Break handler
MOV     AL,23   ; vector for Ctl-C/Break
MOV     DS,DX
MOV     DX,0137 ; point to break routine
INT     21
PUSH    AX      ; clear screen
MOV     AX,2
INT     10
POP     AX
MOV     SI,81   ; check for parameters
LODSB           ; load byte at 81h
CMP     AL,0D   ; see if it is a return
JZ      0135    ; exit to DOS if it is a return (no parameters)
MOV     AH,9    ; print msg
MOV     DX,0148 ; move DX to msg
INT     21      ; function call
MOV     SI,82   ; point to command line parameter
LODSB           ; get the first byte
CMP     AL,0D   ; check to see if it is a return
JZ      0135    ; if so, exit to DOS
MOV     BH,AL   ; put the byte in the BH register
MOV     AH,8    ; call for keyboard input
INT     21      ; function call
SUB     BH,AL   ; compare BH to keyboard input
JNZ     0121    ; if they don't match, start over
JMP     0124    ; otherwise get another byte
INT     20      ; exit to DOS when finished
MOV     AH,9    ; print Ctl-Break msg
MOV     DX,015F
INT     21      ; function call
MOV     AH,7    ; call for keyboard input to return
INT     21      ; function call
CMP     AL,0D   ; check for return
JNZ     0137    ; loop if return is not entered
JMP     010B    ; loop back to program with return
DB 'Please enter passkey: $'
DB 'Sorry, access denied with Ctrl-C and Ctrl-Break. Hit Return $'

RCX
175
N HALT.COM
W
Q
```

End

First, add this line to your CONFIG.SYS file if it isn't already there:

```
DEVICE = ANSI.SYS
```

Make sure that the ANSI.SYS file is in your hard disk root directory and on any self-booting floppy you wish to protect. If you don't have a CONFIG.SYS file (and you should), you can create it with the COPY CON command as follows:

```
COPY CON:CONFIG.SYS
DEVICE = ANSI.SYS
FILES = 20
BUFFERS = 20
BREAK = ON
```

After you type in the listing, press Ctrl-Z or F6 to write the files to disk. You actually don't need the other lines in the file—just the DEVICE line—but the others should improve performance on a typical XT-compatible system with a hard drive.

Next, add the following as the first line in your AUTOEXEC.BAT file:

```
PROMPT $e[37;40m
```

to boot your display to white on black. If you have a color monitor and want other colors, consult your manuals.

Give your AUTOEXEC.BAT a new name (NEWEXEC.BAT, for example) and create a new AUTOEXEC.BAT with any ASCII word processor, Edlin, DOS 5.0's Edit, or the COPY CON command by typing:

```
COPY CON: AUTOEXEC.BAT
ECHO OFF
CLS
ECHO ENTER PASSKEY:
PROMPT $e[30;40m
```

After you type the listing, press Ctrl-Z or F6, and press Enter to write the files to disk. Reboot the disk. If you entered everything correctly, you'll see only the "Enter Passkey" message. Your commands and programs will work, but you won't be able to see anything at DOS level. If you enter the name of an application, such as a spreadsheet, it will load and display it normally. But when you exit and return to DOS, the screen will be blank.

To restore your screen and finish your normal booting procedure with whatever commands were in your AUTOEXEC.BAT file (now the NEWEXEC.BAT file), type NEWEXEC, or whatever name you substituted for AUTOEXEC.BAT. This is your passkey. The first line of your NEWEXEC.BAT file should be:

```
PROMPT $e[37;40m
```

This brings back your normal black-and-white display.

If you want a higher level of security that works with DOS 2.x, type in Program Listing 2 and name the file PASS.SCR. (When typing in the listing, be sure to press Enter after typing the *q* on the last line; otherwise, the program may hang up on you. Do not type the semicolons or the commands that follow them. Use a text editor, such as Edlin, DOS 5.0's Edit, or a word processor that can save files in text or ASCII format.)

Next, copy the file PASS.SCR into the same directory as DEBUG.COM and enter:

```
DEBUG <PASS.SCR
```

Program Listing 4. The DEBUG script file, HIDE.SCR, creates a program that changes the file attributes so that the file does not appear on a normal directory.

```
A 100
MOV     SI,80    ; point to file name length
LODSW            ; load length into CX register
MOV     DX,82    ; point to file name on command line
MOV     DI,82
MOV     AL,0D
SCASB
JNZ     010C
MOV     BYTE PTR [DI-1],0
MOV     AH,43
MOV     AL,01
MOV     CX,02
INT     21
MOV     DX,0125
MOV     AH,09
INT     21
INT     20
DB 'This file is now hidden. DO NOT FORGET the name! $'

N HIDE.COM
RCX
60
W
Q
```

End

Program Listing 5. The DEBUG script file, SHOW.SCR, creates a program that lets the file or subdirectory reappear on a normal DOS directory.

```
A 100
MOV     SI,80    ; point to filename length
LODSW            ; load length into CX register
MOV     DX,82    ; point to filename on command line
MOV     DI,82
MOV     AL,0D
SCASB
JNZ     010C
MOV     BYTE PTR [DI-1],0
MOV     AH,43
MOV     AL,01
MOV     CX,0
INT     21
MOV     DX,0125
MOV     AH,09
INT     21
INT     20
DB 'This file can now be seen in the directory. $'

N SHOW.COM
RCX
60
W
Q
```

End

If you see no error messages displayed as DEBUG assembles your program, then type PASS at the DOS prompt. The screen should clear, and you should see the password prompt at the top of your screen. Your passkey is an ampersand (&) in this case, but you can change it to anything you want by changing the hex number in the second line (26 hex represents the ampersand key).

If you type PASS as the first command in your AUTOEXEC.BAT file, the password prompt will appear whenever you start the system. The computer will not respond to Ctrl-C or Ctrl-Break, so you cannot break out of the program except to enter the proper passkey or to reboot. Rebooting, however, brings you back to the same place.

PASS.SCR is an extremely simple program offering some protection against unwarranted access to your system. A person randomly pressing keys eventually will press the right one and pass through. However, you can rewrite PASS.SCR to require two, three, or any number of passkeys, even if you are only slightly familiar with assembly language programming.

Higher Levels of Protection

Program Listing 3 is a more sophisticated program that lets you supply a passkey up to 118 characters long. The passkey can be virtually anything, even a quotation. Type the listing into an ASCII file named HALT.SCR and use DEBUG to assemble it:

```
DEBUG <HALT.SCR
```

To use the program, type HALT followed by a space. Then enter a passkey and press enter. The screen should clear and display the password prompt. To pass through, type in your passkey. Nothing you type is displayed on screen (so that no one looking over your shoulder can see the passkey). If you make a mistake, type the passkey again without pressing enter. As with the PASS.SCR program, HALT.SCR is case-sensitive and does not respond to Ctrl-C or Ctrl-Break. You must type in your passkey exactly as you did the first time using the same capital and lowercase letters. If you enter the HALT command without a passkey, the program displays the DOS prompt.

For an even higher level of security, you might consider other options. For example, you could rename the

TYPE Command in the same way that you renamed the COPY, DELETE, and ERASE commands. You also could try giving your files names that disguise their contents. The latter option might fool an intruder who is looking for a specific file in your system.

Another option is to change the file attributes of sensitive files. The DEBUG script file called HIDE.SCR (see Program Listing 4) changes the file attribute so that the file does not appear on a normal directory. HIDE.SCR also works for subdirectories, but it does not accept wildcard characters (* or ?). Use DEBUG to assemble the program:

```
DEBUG <HIDE.SCR
```

You should be aware, however, that even hidden files may appear in many directory utilities, DOS shell directories, and when you invoke TREE.COM.

The DEBUG script in Program Listing 5, which you enter into an ASCII file called SHOW.SCR, reverses the HIDE.SCR process so that the file or subdirectory reappears on a normal DOS directory. Use DEBUG to assemble the program:

```
DEBUG <SHOW.SCR
```

The listing is identical to HIDE.SCR except for the name, the message, and the hex number in the line that reads, "MOV CX." In HIDE.SCR, 02 hex in the CX register changes the attribute to hidden. In SHOW.SCR, the zero in the CX register changes it back so that it can be seen on a DOS directory.

Another method of file protection is to scramble your files so that they cannot be read directly from the console display. No scrambling technique is foolproof, but the scramble program SCRAMBLE.BAS (see Program Listing 6) makes your sensitive text files unreadable to everyone but those adept at ciphering.

Program Listing 6. SCRAMBLE.BAS is a program that scrambles ASCII files so they are not readable from the console.

```
10 INPUT "Enter name of file to scramble: "; N$
15 INPUT "Enter name of file to write scrambled data: "; O$
20 OPEN "I", 1, N$
25 OPEN "O", 2, O$
30 IF EOF(1) THEN 70
35 A$ = INPUT$(1, #1)
40 IF ASC(A$) <= 127 THEN GOTO 60
45 PRINT : PRINT "*** Can't Scramble. File ‘"; N$; "‘ is not a text file."
50 CLOSE #1, #2: NAME O$ AS "killfile": KILL "killfile"
55 GOTO 75
60 PRINT #2, CHR$(ASC(A$) + 128);
65 GOTO 30
70 CLOSE #1, #2: NAME N$ AS "killfile": KILL "killfile"
75 PRINT
80 END
```

End

Program Listing 7. DESCRAM.BAS unscrambles text files scrambled by SCRAMBLE.BAS.

```
10 INPUT "Enter name of file to descramble: "; N$
15 INPUT "Enter name of file to write descrambled data: "; O$
20 OPEN "I", 1, N$
25 OPEN "O", 2, O$
30 IF EOF(1) THEN 75
35 X$ = INPUT$(1, #1)
40 Y = ASC(X$) - 128
45 IF Y > 0 GOTO 65
50 PRINT : PRINT "*** Can't Descramble. File ‘"; N$; "‘ is not a Scrambled file."
55 CLOSE #1, #2: NAME O$ AS "killfile": KILL "killfile"
60 GOTO 80
65 PRINT #2, CHR$(Y);
70 GOTO 30
75 CLOSE #1, #2: NAME N$ AS "killfile": KILL "killfile"
80 PRINT
85 END
```

End

SCRAMBLE.BAS works only with text files, so don't use it with .COM, .EXE, or other kinds of binary files.

SCRAMBLE.BAS uses a simple substitution scheme. It reads the data (usually a character) from your file, adds 128 to each data item, and then writes the data back to a file you specify. Your original file is then deleted. By adding 128 to each data item, you change the item from an alphanumeric character (for text files) to a control character that appears as a meaningless symbol.

The program DESCRAM.BAS reverses the process (see Program Listing 7). These programs work as listed with both GW-Basic and DOS 5.0's QBasic.

A Bit of Common Sense

The methods and programs described in this article may be the only protection most PC users need, but remember, if your files are sensitive and you can't be around to guard them, the best way to protect them is to remove them from your hard disk. Take your floppy disks with you or safely lock them away, and always observe the safety and security features of any network on which you participate. In the final analysis, PC security is a matter of common sense. ■

Larry Sessions is a free-lance writer whose articles have appeared in Science Digest *and* Rolling Stone.

Questions & Answers

WINDOWS 3.0

WHY WINDOWS?

We can't avoid it, and try as you might, you can't either. Windows is the issue, that friendly computing face with a vice-like handshake. Sooner or later, all DOS users will be exposed to the Windows operating system from Microsoft.

Our goal at *DOS Resource Guide* is to remain true to our prompt, but as one computer magazine columnist pointed out, the Windows graphical environment is so attractive to software publishers that soon the best new software for the PC-compatible world will be available *only* under Windows. That's a pity, but the insight probably will prove true.

One estimate in the trade press asserted that 50 books had been issued on Microsoft's Windows, and the obvious question is why? Windows, as advertised, is the operating system for the "rest of us." Full of Macintosh-like icons (file folders represent directories; an hourglass tracks the time you are wasting), Windows has been heralded as an operating system that will make even a neophyte computer owner a productive worker in a matter of hours.

But that's not quite true. Windows is neither "intuitive," as its maker claims, nor is it—at first—very easy to use. Windows is also a hungry program that demands extra hardware and memory (probably a 386SX computer or better and at least 2MB of RAM), and even then it may cause you configuration headaches and memory conficts. DOS's own APPEND, FASTOPEN, and SHARE commands—typically invoked in one of your system's configuration files— aren't on the Windows diet. Windows, itself, has a variety of modes in which it can run—real, standard, or 386. But when should you use one mode instead of another?

Finally, of course, Windows requires its own software. As a rule, these programs are more expensive than DOS software, and they, too, are a mixed blessing. Designed to work from a similar interface, each program works with its own set of icons and rules.

Because there is no getting around the complexity of the transition to Windows for the DOS user, *DOS Resouce Guide* sought the help of an expert for your questions. ***Brian Livingston***, author of the best-selling book *Windows 3 Secrets,* (IDG Books Worldwide, San Mateo, Calif., $39.95), has been answering questions about Windows for a number of years. His book is a study of tricks and undocumented tips about how to get around in and live happily with this colorful DOS add-on. He'll be writing regularly on the topic of running Windows.

Q My computer's AUTOEXEC.BAT file contains a line that loads DOS 5.0's expanded memory manager, EMM386.EXE. I use EMM386.EXE's RAM parameter to claim memory in the upper memory blocks area so that DOS can move my system's device drivers and TSRs out of conventional memory into upper memory. But when I attempt to do this, Windows won't start up in standard mode. Can you tell me what the source of the problem is?

A An incompatibility exists between EMM386.EXE and Windows 3.0. This incompatibility also applies to the older version of EMM386.EXE that comes with Windows 3.0, EMM386.SYS. If you load either of these memory-management programs and use any of the expanded memory they provide, Windows will not start up in standard mode (it will, however, work in real mode and enhanced mode). To solve the problem, you must use a third-party memory manager, such as Quarterdeck Office Systems' QEMM386.SYS, or wait for the release of Windows 3.1, which should cure the problem.

Q When should I use the /Y parameter with MOUSE.SYS? What about MOUSE.COM?

A The /Y switch forces graphics adapters, such as Video Seven boards, to display a mouse cursor as a small, square block character in DOS applications that run in text mode. Without the /Y parameter (which stands for, *Yes, I'm using a Video Seven-type graphics board*), these adapters display a mouse cursor as a graphical character called a sprite. This graphical character can cause Windows to become confused because it expects DOS applications to contain only text characters. It's safer to use the /Y switch, which Windows automatically adds to MOUSE.SYS if you choose to have Windows Setup place MOUSE.SYS in your CONFIG.SYS file. The same is true of MOUSE.COM, which you might place in your AUTOEXEC.BAT file to provide mouse support for DOS applications.

Q While using a disk utility, I recently discovered a large hidden file called 386SPART.PAR. Not long ago, I installed Windows 3.0. Is this file part of Windows?

A 386SPART.PAR is a "permanent swap file," a file Windows uses to copy programs to disk when it runs out of room in RAM. You also may find a read-only file in your Windows directory named SPART.PAR, which serves as an index to the 386SPART.PAR file. These SPART files (*SPART* stands for *special partition*) reserve one continous block of your hard disk for the exclusive use of Windows. Windows can write more quickly to the 386SPART.PAR file than it can to other areas of your hard disk because 386SPART.PAR has a special format that is unlike that of normal DOS files.

Deleting, moving, or renaming either of these two files by typing a command at the DOS prompt will create problems. If you want to remove them or make your permanent swap file a different size (perhaps because you need free hard disk space for other programs), use the SWAPFILE.EXE utility included with Windows. To do this, start Windows in real mode, using the command:

```
WIN /R
```

Close any applications that might be running, except for your Windows shell—Program Manager, perhaps. Then, select the File menu's Run option and type:

```
SWAPFILE
```

in response to the prompt for a filename and press Enter or click on OK. The Swapfile dialog box appears, and you can delete or resize the permanent swap file.

Program Listing 1. WIN.BAT provides reminders to prevent you from accidentally running Windows a second time or ending a computing session when applications might be running.

```
ECHO OFF
IF "WIN"=="Y" GOTO :INWIN

  SET WIN=Y
  SET HOLDER=%PROMPT%
  SET PROMPT=Type EXIT to return to Windows.$_$P$G
  WINDOWS
  SET WIN=
  SET PROMPT=%HOLDER%
  GOTO :END

:INWIN

  ECHO You are already running Windows.
  ECHO Press Alt-Tab to switch to Windows
  ECHO or type EXIT to return to Windows.

:END
```

End

Q How much memory do I really need to run Windows on a 286 machine? Or should I even bother trying to run Windows on a 286?

A If you run only simple programs—perhaps using Notepad to type short text files—you may find that Windows runs fast enough on your 286 to keep up with your typing. But if you use more complex applications, such as a word processor or spreadsheet, you may spend more time waiting for Windows to run than you do getting work done.

Two primary factors affect Windows' performance on a 286: CPU speed and available memory. Windows is very slow on older 6MHz and 8MHz 286s. But you might find acceptable performance on newer 12MHz and 16MHz 286s. Even on a 16MHz 286, however, you probably will find that running Windows with less than 2MB of RAM isn't worth the effort. That's because Windows itself claims about 1MB of memory, and any significant application you start is going to need at least some of the remaining 1MB.

Several tests, however, have shown that Windows really runs better with 4MB of memory than with 2MB. This gives Windows enough RAM for both itself and your applications and data. Also, 4MB of RAM lets you set aside 512K to 1MB of memory for a disk-caching utility, such as SMARTDRV.SYS (which comes with Windows). On a 2MB machine, setting aside more than 256K for a disk cache is probably not the most efficient use of your system's memory.

Q Sometimes when I am running a full-screen DOS session from inside Windows, I forget I'm in Windows. Then, I tend to attempt to start Windows a second time or even turn off the computer. Is there a simple way to avoid this?

A Running Windows a second time in a DOS session under Windows is dangerous. It can cause Windows to hang, and some applications lose data if you turn off your computer before you exit the application completely.

For this reason, you should start Windows from a batch file, instead of starting Windows simply by invoking it at the DOS prompt. This batch file can protect you against the peril of starting Windows twice—or starting other programs that are incompatible with Windows.

First, rename your WIN.COM program to WINDOWS.COM, so you can have a batch file called WIN.BAT. (When you type the name of an executable file at the DOS prompt, DOS always looks for a file with a .COM extension before it looks for one with a .BAT extension, which means DOS will never find WIN.BAT if WIN.COM already exists.) If your copy of Windows is in a \WINDOWS directory on drive C, the commands to rename WINDOWS.COM are as follows:

```
C:
CD \WINDOWS
REN WIN.COM WINDOWS.COM
```

Next, use a text editor to create the batch file named WIN.BAT shown in Program Listing 1.

This batch file does two things: It sets an environment variable named Win equal to Y (for *Yes*), and it changes your DOS prompt. Let's examine the commands that do this—I've indented some of the lines to make it easier to see the two major "branches" the batch file can take.

After you invoke the batch file, DOS reads the first line of the file, shuts off echoing to the screen to keep the screen neat and proceeds to the second line. This line tests to see if a variable named Win already exists in the environment. If Win exists and is equal to Y, DOS proceeds to the branch labeled :INWIN. Because you are runnning the batch file for the first time, no such variable exists at this point, so DOS drops down to the next program line. There, the variable Win is set to Y. The next line captures the value associated with your DOS prompt in a variable named HOLDER. The next statement substitutes a new, informative DOS prompt for the old prompt.

The batch file then issues the WINDOWS command to start WINDOWS.COM. Whenever you leave Windows temporarily—by selecting the DOS prompt option in Program Manager—DOS returns you to the batch file, which hasn't yet ended and you see the reminder contained in the prompt. The cryptic sequence $_$P$G tells DOS to display a blank line and then provide the current pathname followed by a greater-than sign (>).

When you quit Windows, the remainder of the batch file runs. The SET WIN= statement eliminates the Win variable by setting it equal to nothing, and the subsequent line stores the contents of HOLDER, your old prompt, in the variable named PROMPT.

If, instead of going about your business in Windows and quitting at the end of your session, you forget Windows is running and try to run it again, the batch file will run again. This time, the value of Win will be Y (because you haven't yet quit the program and DOS therefore hasn't yet reset the value of Win to nothing), and DOS will drop down to the :INWIN label. This part of the batch file echoes a message to the screen, reminding you that you are in Windows and telling you what to type to return to Windows.

If you prefer, you can choose not to change WIN.COM's name and call your batch file W.BAT, instead. Then, you simply need to remember to start Windows by typing:

```
W
```

or by placing this command at the end of your AUTOEXEC.BAT file to start Windows every time you turn on your PC.

Program Listing 2. This sample batch file shows how you might use tests for environment variables to avoid executing DOS programs that should not be run under Windows.

```
ECHO OFF
IF "WIN"=="Y" GOTO :DONOTRUN

  DISKOPT
  GOTO :END

:DONOTRUN

  ECHO You are running under Windows.
  ECHO Type EXIT to return to Windows,
  ECHO then quit Windows before using
  ECHO this disk optimizer program.

:END
```

End

Program Listing 3. This sample batch file reminds you not to run DOS from within 1-2-3 when Windows is in charge.

```
ECHO OFF
SET HOLDR123=%PROMPT%
  SET PROMPT=Type EXIT to return to 1-2-3.$_$P$G
IF NOT "WIN"=="Y" GOTO :RUN123
  SET PROMPT=You shouldn't shell to DOS while Windows is running.$_$P$G
 :RUN123
  123
  SET PROMPT=%HOLDR123%
```

End

You can use a similar technique to that shown in WIN.BAT to protect yourself from executing a program that should not be run in a DOS session under Windows. You should not, for example, run disk backup or disk optimizing programs while you are working under Windows because doing so might inadvertantly scramble files that DOS is using.

It's best to run utilities such as these from their own batch files, which test for the variable Win before starting the dangerous operation (see Program Listing 2).

Like the previous batch file, this one tests for a variable called Win. If it finds it, and it is equal to Y, the batch file jumps to the :DONOTRUN label and displays a message explaining what to do next.

You also can use this technique to alter the DOS prompt for other applications that let you shell to DOS. In Lotus 1-2-3, for example, you can get back to the DOS prompt by selecting the System menu option. Because of Windows' layered architecture, applications that you run by shelling to DOS usually are much slower than if you switch back to Windows by pressing Alt-Tab and starting a separate DOS session from Windows. Program Listing 3 shows how you might remind yourself to start a DOS session from your 1-2-3 icon.

This batch file saves your original prompt, then sets one of two possible prompts. The batch file first sets the prompt to one that is appropriate for DOS sessions running under 1-2-3. If Windows isn't operating, the batch file jumps ahead and starts 1-2-3 immediately. If, on the other hand, Windows is running, the batch file then resets the prompt to a message that discourages you from shelling to DOS under 1-2-3 while Windows is running.

Of course, you can make your batch files much more complex than this, but these three examples provide a good starting point.

Q I know Windows 3.0 can benefit from a RAM drive and a disk cache. On a 386SX with 2MB of memory, how much should I allocate to each?

A Using RAM drive and disk-caching utilities can improve the performance of Windows on almost any PC. But 2MB is about the minimum amount of memory you should have before you start allocating memory to these utilities. Otherwise, you reduce the amount of memory Windows can use for itself.

On a 2MB PC, one rule of thumb is to give a disk cache 1MB of RAM when only DOS applications are running and to reduce this to 512K when Windows is running. The reason for this difference is that, unlike Windows, most DOS applications cannot use extended memory so they need a larger disk cache.

If you use SMARTDRV.SYS, the disk cache included with Windows, you would implement this "1MB vs 512K" rule by putting the following line in your CONFIG.SYS file:

```
DEVICE=C:\WINDOWS\SMARTDRV.SYS
  1024 512
```

(This example assumes that SMARTDRV.SYS is in the \WINDOWS directory on drive C.) If you have more memory, you can allocate 2MB to SMARTDRV when your system is running outside Windows, and 1MB when Windows is running.

RAM drives do not improve the performance of Windows nearly as much as a disk cache does. In fact, one of the few ways that a RAM drive can improve Windows performance is speeding up the temporary file that Windows' Print Manager makes before it prints a file. Since a single page of a print file with graphics can contain as much as 1.5MB of data, you must have a RAM disk of at least 2MB to ensure that Print Manager will have enough room to benefit from the RAM drive.

If you have 6MB or more of RAM, and it is configured as extended (not expanded) memory, you might want to make a 2MB RAM disk. You can do this using RAMDRIVE.SYS, the RAM drive utility included with Windows 3.0. Place the following line in your CONFIG.SYS:

```
DEVICE=C:\WINDOWS\RAMDRIVE.SYS
  2048 /E
```

The /E switch in this line forces the RAM drive utility to use extended memory, which is faster than expanded memory.

For Print Manager to find and use this RAM disk for its temporary files, however, you must correctly set a variable named Temp. You can do this in your AUTOEXEC.BAT file. If your main hard disk is C, and your RAM disk is named D (DOS uses the drive letter of the next available logical drive), you can insert the following lines:

```
SET TEMP=C:\
IF EXIST D:\NUL SET TEMP=D:\
```

The first line sets the Temp variable to the root directory on drive C; the subsequent line resets it to the root of your RAM drive—if one exists. (If a RAM drive doesn't exist, DOS proceeds to the next line of your AUTOEXEC.BAT.) The second statement tests for the existence of DOS's imaginary NUL device, which exists theoretically in every disk directory you have. Therefore, if D:\NUL exists, then the RAM drive D must also exist, and the statement makes this the location of Print Manager's temporary files.

To see how this setup works, first use the Control Panel's Printers dialog box to ensure that your Print Manager is turned on. Then start File Manager, copy a small dummy file to your RAM drive, and open a directory window for the root directory on drive D. Finally, open a window for another application, and position the second window so that you can see both windows at the same time. When you print a medium-size file from your application, you should see a temporary file created and then deleted from D:\—a sign that Print Manager is at work.

Q When I'm running a non-Windows (DOS) application in a window, can I use a mouse with the DOS application?

A You can use a mouse in a DOS application under Windows only if the DOS application occupies the entire screen. When you press Alt-Enter to switch the DOS application into a window (which is possible only in 386 enhanced mode), Windows grabs the use of the mouse for itself—so that it can point to other icons on the screen, for example.

But you can do one very handy thing with the mouse while your DOS application is windowed: You can use it to transfer text from the DOS application into the Clipboard, and from there into any Windows application—even if your DOS application doesn't support the Clipboard.

To do this, place your DOS application in a window. Then click your mouse anywhere in that window. The title of your DOS window should change from something such as MYDOSAPP to Select MYDOSAPP. This means that any information in the DOS application that you drag your mouse over is being selected for transfer to the Clipboard.

Once you select a rectangular area of text (you'll see the area change color to indicate what you've selected), press Enter or click the second mouse button (the right button, if you have your mouse configured for right-handed use). This copies the text into the Clipboard and returns your DOS application to normal. If you open the Windows Clipboard, you'll see a copy of your DOS text. Then open any Windows word processor, place the cursor where you want the text, and click on the Edit Paste option. Presto! The text is placed right where you want it.

If your DOS application uses IBM PC line drawing characters (such as | |) to box information, Windows may not display these characters properly. You can click on Display OEM-Text in the Clipboard to make Clipboard show them correctly. But this doesn't mean that Windows word processors will import them from the Clipboard as expected. You'll probably have to delete such characters yourself.

Q I'm running a non-Windows communications package (Procomm Plus) under Windows 3.0 on a 16MHz 386SX. If I start a file transfer with Procomm Plus occupying a full screen, then use Alt-Enter to put it into a window (so I can play Solitaire while the transfer proceeds), everything grinds to a halt. My hard drive light comes on, but it takes several minutes for Procomm Plus to switch into a window, and the transfer often times out. Even if the transfer does not time out, every action—even such tasks as changing the active window or pulling down a menu—is excruciatingly slow while the transfer is in progress. Am I expecting too much of Windows?

A Several factors can adversely affect Windows' response time while a program is performing communications in the background. Reducing to a window a DOS program that occupies an entire screen requires more memory than is necessary simply to start the program. If Windows is short of memory, this switch can force Windows to swap a few Windows applications to disk to free up memory for the windowed DOS application. One way to avoid the problem you're having is to switch Procomm Plus into a small window *before* you establish communications. Procomm should still be able to dial out, even though it is running in a window. Remember that you cannot use a mouse to do anything in a DOS application while it is windowed.

Downloading a file can take up a large percentage of the CPU time on a 386SX, another possible explanation for the slowness of Windows applications while Procomm Plus is working. But it's also possible that you have the settings for background processing in the program information file (PIF) for Procomm Plus set too high.

You should run DOS applications from a PIF file. If you don't have one for Procomm Plus, make one using Windows' PIF Editor, call it PROCOMM.PIF, and change your Procomm Plus icon so it runs that PIF.

Check the Advanced options in your PROCOMM.PIF. These options include the priority settings this application will receive when running in the foreground and in the background. These settings are usually set to 50 timer units for a DOS application in the background, while Windows gets 100 units when it is in the foreground (the setting for Windows is determined in the Control Panel's 386 Enhanced dialog box). Try changing the background priority of Procomm Plus from 50 to 25. Then try a setting of 10 to see if Procomm Plus still works. If it does, your Windows applications should be more responsive.

Finally, you may need a better serial interface board to handle file transfers in the background and still get acceptable performance from Windows. Serial boards such as the Hayes ESP board, available from most computer dealers, fit this need. Boards such as this use the newer 16550 communications chip and have a 16-byte buffer in which to store incoming communications data while Windows is busy handling other tasks. Sixteen bytes may not sound like much, but at the speeds of a 386, it can make a tremendous difference in communications throughput. Use of this buffer requires that you replace Windows' COMM.DRV driver with a specialized communications driver such as TurboCom, which is included with the Hayes ESP board.

Q How do I make expanded memory available to a DOS application running under Windows?

A Two methods exist for giving DOS applications access to expanded memory under Windows: One works in 386 enhanced mode; the other in real and standard modes.

In real and standard modes, you must have a third-party expanded memory management utility (such as QEMM386.SYS) running before you start Windows. Then, when you start your DOS application under Windows, it can find the available expanded memory and use it.

In 386 enhanced mode, Windows has the ability to convert extended memory into expanded memory on demand for use by DOS applications. Simply configure all your memory chips as extended memory, and let Windows respond automatically to DOS applications that are running under Windows and need expanded memory.

You must ensure that the PIF file that starts the DOS application provides enough expanded memory in 386 enhanced mode for the DOS application to run. Use the PIF Editor, in enhanced mode, to check on the amount of expanded memory allowed to the application. I always set the PIF so that at least 256K of expanded memory is "required," with an upper limit of at least 1536K. These settings usually result in your application gaining access to about 1024K of expanded memory (since not all of the 1536K of expanded memory that Windows manages may become available to the application). If you have enough memory, you can afford to increase these settings. And do turn on the "locked" box in the PIF, so no other application can take away expanded memory once your DOS application has started using it.

Q What's the best way to configure my computer's memory before I install Windows?

A Configuring the memory chips in your PC *before* you install Windows is a good idea, because Windows' Setup program then can allocate memory automatically—for such utilities as SMARTDRV.SYS.

Configuring your memory largely involves deciding how to divide your memory chips between two types of memory: expanded and extended memory. Another type of memory, conventional memory, almost always has exactly 640K available, so no decision is necessary about that memory.

In real mode, Windows can use only expanded memory. In 386 enhanced mode, Windows uses only extended memory, not expanded memory. Therefore, if you use one of these two modes exclusively, configure your memory chips to provide the type of memory that Windows uses. Most DOS applications do not use extended memory, although some do use expanded memory. But on a 386, Windows and other third-party memory-management utilities can convert extended memory into expanded memory on demand. So on a 386, extended memory is almost always the preferred configuration.

If you use Windows in standard mode, you need to make some decisions. Windows in standard mode uses only extended memory. It cannot use expanded memory nor can it convert extended memory into expanded memory for DOS applications. Therefore, you should divide your memory chips into enough extended memory for Windows' use, and enough expanded memory for your DOS applications' use. You can decide how much expanded memory you need by loading your largest document for the DOS application that requires expanded memory. Then, use that program's memory use option to compare how much expanded memory is available and how much is in use. You can convert any unused expanded memory to extended memory. Windows can use up to 15MB of extended memory.

Your PC or expanded memory board manual should include instructions on how to set your software or switch settings to divide your memory between expanded and extended memory. ■

Brian Livingston is the author of Windows Secrets *(IDG Books Worldwide, San Mateo, Calif.).*

Tip 9

Conquering Space

Normally, DOS does not allow you to include spaces in a filename. Unfortunately, some companies take advantage of this quirk during software installation procedures, creating files you can't modify using standard DOS commands. Suppose, for example, that you want to rename a file called OUR LIST.DAT, which was created by an application during installation. If you try to do this simply by typing:

```
REN OUR LIST.DAT DATA.HLD
```

DOS returns a "Too many parameters" error message. Replacing the blank space with the DOS ? wildcard makes the change possible:

```
REN OUR?LIST.DAT DATA.HLD
```

Subscribe to GamePro! The Premiere Video Game Magazine.

- ★ Your magazine for Nintendo, Sega, Atari, TurboGrafx, Genesis, and GameBoy.
- ★ Tons of tips, strategies, and secret moves to help you beat your best.
- ★ In-depth reviews on new and pre-released games to keep you in the know.
- ★ Reviews of the hottest games from overseas.
- ★ The latest info on cutting-edge technology.
- ★ The Adventures of GamePro–follow our comic hero through the Video Dimension!
- ★ An exciting, dynamic look that will knock your socks off!

12 Issues Only $19.97.

Over 56% off the cover price.

Name

Address

City State Zip

Age Game System(s)

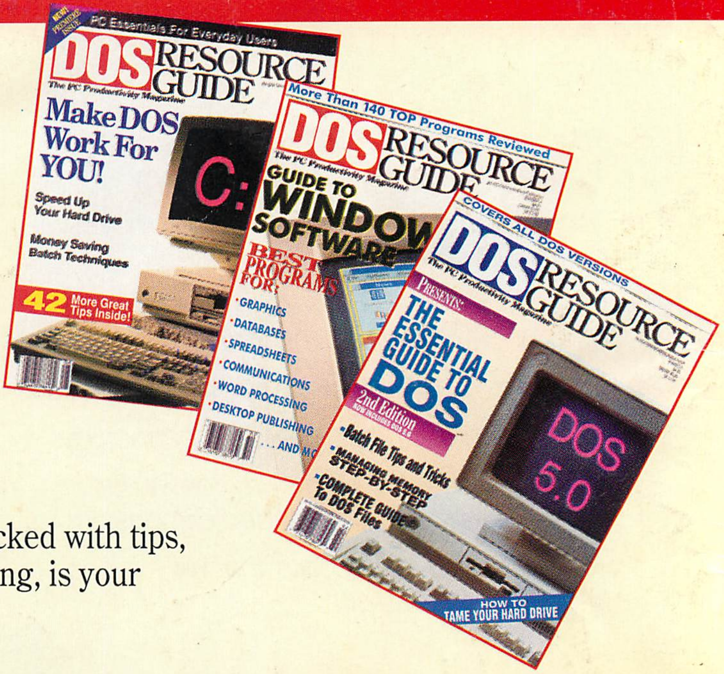
Credit Card No. (VISA/Master Card only) Exp. Date

Name on Card Signature

Place your check of money order, payable to GAMEPRO, in an envelope and mail to:
GAMEPRO Magazine, Circulations, P.O. Box 883, Farmingdale, NY 11737
No cash please. Offer valid in USA. For Canada add $10.00. Allow 6-8 weeks. 7181GP

NINTENDO • SEGA • TURBOGRAFX • GENESIS • GA
GAMEPRO
January 1990
An IDG Communications Publication
Cutting Edge Action!
Check Out The New TurboGrafx CD Player
Blast Out The New Year With Hot New Hits!
Double Dragon II
SpellCaster
Golden Axe
Dungeon Explorers
Tips, Tips, & More Tips!
A Classic!
Relive The Awesome
Dragon
NINTENDO • SEGA • TURBOGRAFX • GENESIS • GAMEBOY •
GAMEPRO
December 1989
An IDG Communications Publication
Incredible! Amazing!
Blow Your Mind on Our Holiday Issue!
28 Pages of New Games You've Just Gotta Have
Rolling Thunder
Dynamite Dux
The Last Battle
Alien Crush
Tips & Tactics
Sharpen Your Skills With These Wild Moves!
Free Poster!
See Page 4
$3.95 Canada $4.95
Exclusive!

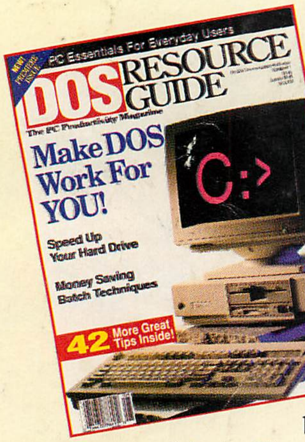
TAKE THE DREAD OUT OF DOS!


DOS RESOURCE GUIDE
Make DOS Work For YOU!
GUIDE TO WINDOWS SOFTWARE
THE ESSENTIAL GUIDE TO DOS
HOW TO TAME YOUR HARD DRIVE

If you're trying to succeed with DOS, but feel like a General Failure...

DOS RESOURCE GUIDE IS WRITTEN FOR YOU

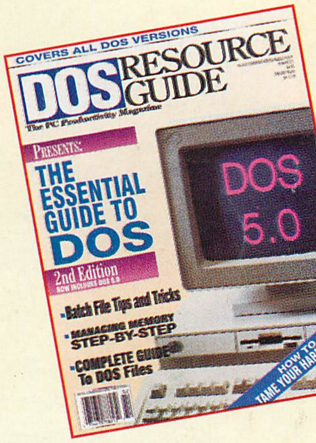
DOS RESOURCE GUIDE, the quarterly periodical packed with tips, tutorials, and information on all aspects of PC computing, is your antidote for computing confusion.



DOS RESOURCE GUIDE #1

Make DOS Work for You

Covers six areas of computing that are vital to DOS users: start-up files (AUTOEXEC.BAT and CONFIG.SYS), batch file creation, hard drives, safe backups, organizing file management, and hardware maintenance. Includes: List'em, a batch file that helps you keep appointments; 10 tips to keep your hard drive healthy; speed-up tips for a dogged hard drive; a batch file that automates backups; DOS shortcuts, hardware dos and don'ts. Plus 42 provocative pointers in DOS.

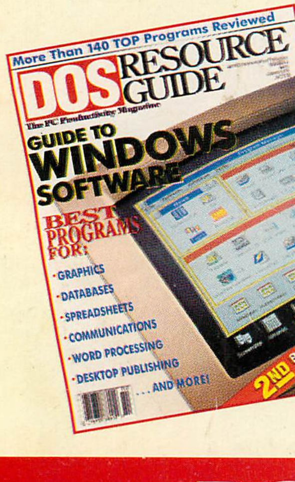

DOS RESOURCE GUIDE
THE ESSENTIAL GUIDE TO DOS
2nd Edition
Batch File Tips and Tricks
MANAGING MEMORY STEP-BY-STEP
COMPLETE GUIDE To DOS Files
HOW TO TAME YOUR HARD DRIVE

DOS RESOURCE GUIDE #3

The Essential Guide to DOS

Who are we to argue with success! Our second edition of this classic is updated to include information on DOS 5. Guide #3 includes an annotated list of DOS files and when to use them; articles that will change your view of CHKDSK, COPY, and XCOPY; a cluster of batch file assists; and some clear-headed advice on organizing files on your hard drive. Plus 14

So, If DOS keeps asking you those hair-raising questions, like "Are you sure?"... THEN take a look at a few back issues of DOS RESOURCE GUIDE, and take the dread out of DOS!



DOS RESOURCE GUIDE #2

Guide to Windows Software

A compendium of software briefs that tells you what's up in the world of Windows. (Even a DOS user has to keep up with the competition.) Guide #2 briefly reviews over 140 programs for Microsoft Windows including word processors, communications programs, financial and mathematical programs, management programs, developer's tools, and graphics software. Complete list of manufacturers.

Use our 800 number and have your credit card ready!

1-800-343-0728

Back issues of DOS RESOURCE GUIDE are $5.50 post-paid!

Or send your check or money order to:

DOS Resource Guide
Back Issues
80 Elm Street
Peterborough, NH 03458

(Please indicate what issue(s) you are purchasing.)

DON'T MISS AN ISSUE!